10-11-2020

Pro developers en PowerApps con Power Apps Component Framework

Microsoft 365 Saturday Madrid Virtual 2020

Contenido

[**Ejercicio 1: Desplegar la solución base en nuestro entorno** 5](#_Toc55500902)

[**Ejercicio 2: Crear un menú global y una pagina de detalle de sesión con Bibliotecas de componentes** 11](#_Toc55500903)

[**1.** **CREACIÓN DE UN MENU CÓMUN CON COMPONENTES** 11](#_Toc55500904)

[**2.** **CREACIÓN DE UN MENU CÓMUN CON COMPONENTES** 15](#_Toc55500905)

[**3.** **INTEGRAR COMPONENTES EN EL APP PRINCIPAL** 17](#_Toc55500906)

[Integrar el componente Menu 18](#_Toc55500907)

[**Intregrar el componente “Detail Session”** 19](#_Toc55500908)

[**Ejercicio 3: Crear un componente con Power App Component Framework en React con VS Code** 20](#_Toc55500909)

[Preparar la aplicación 20](#_Toc55500910)

[**Desarrollar el componente Agenda con Power App Component Framework** 21](#_Toc55500911)

[Depurar en local y probar su ejecución 29](#_Toc55500912)

[**Desplegar el componente como una solución en Power Apps** 30](#_Toc55500913)

[Subir la solución a nuestro entorno de Test 31](#_Toc55500914)

Introducción

Tiempo estimado para completar este laboratorio

90 minutos

Prerrequisitos

Para este laboratorio se requiere una licencia de Power App Plan, ya que se hace uso de servicios Premium.

Para ello podemos crear una suscripción de prueba de 30 días <https://docs.microsoft.com/es-es/powerapps/maker/signup-for-powerapps>

**Instalar Power App CLI**

Para obtener Power Apps CLI, realice lo siguiente:

1. Instale [Npm](https://www.npmjs.com/get-npm)(se incluye con Node.js) o [Node.js](https://nodejs.org/en/) (se incluye con npm). Recomendamos LTS (soporte a largo plazo) versión 10.15.3 o superior.
2. Instale [Paquete de desarrollador .NET Framework 4.6.2](https://dotnet.microsoft.com/download/dotnet-framework/net462).
3. Instale [.NET Core 3.1 SDK](https://dotnet.microsoft.com/download/dotnet-core/current) e instale [Visual Studio Code](https://code.visualstudio.com/Download)
4. Instalar [Power Apps CLI](https://aka.ms/PowerAppsCLI).

Objetivos

Después de haber completado este laboratorio serás capaz de:

* Crear un Componente desde el Power Apps Studio
* Crear un Componente con el Framework de desarrollo
* Integrar componentes con nuestras aplicaciones y datos en CDS
* Manejar al 100% la integración de componentes en Canvas App
* Empaquetar soluciones, desplegar e instalar en nuestros entornos
* Introducción a gestión de soluciones y entornos

Descripción del Laboratorio

La sesión va a consistir en desplegar en nuestro entorno la solución “Gestión de eventos”, y evolucionarla mediante la construcción de diferentes componentes.

La solución se basa en una solución Canvas en Power App, y una serie de entidades en CDS donde alojaremos los datos de la app, que podremos analizar en profundidad durante la sesión.

Al ser CDS todos los datos de inserción en tabla, lo haremos mediante los Formularios de la propia entidad, y la Aplicación en esta versión solo será consultiva.

Gestión de eventos

La aplicación se basa en la visualización de los eventos pudiendo:

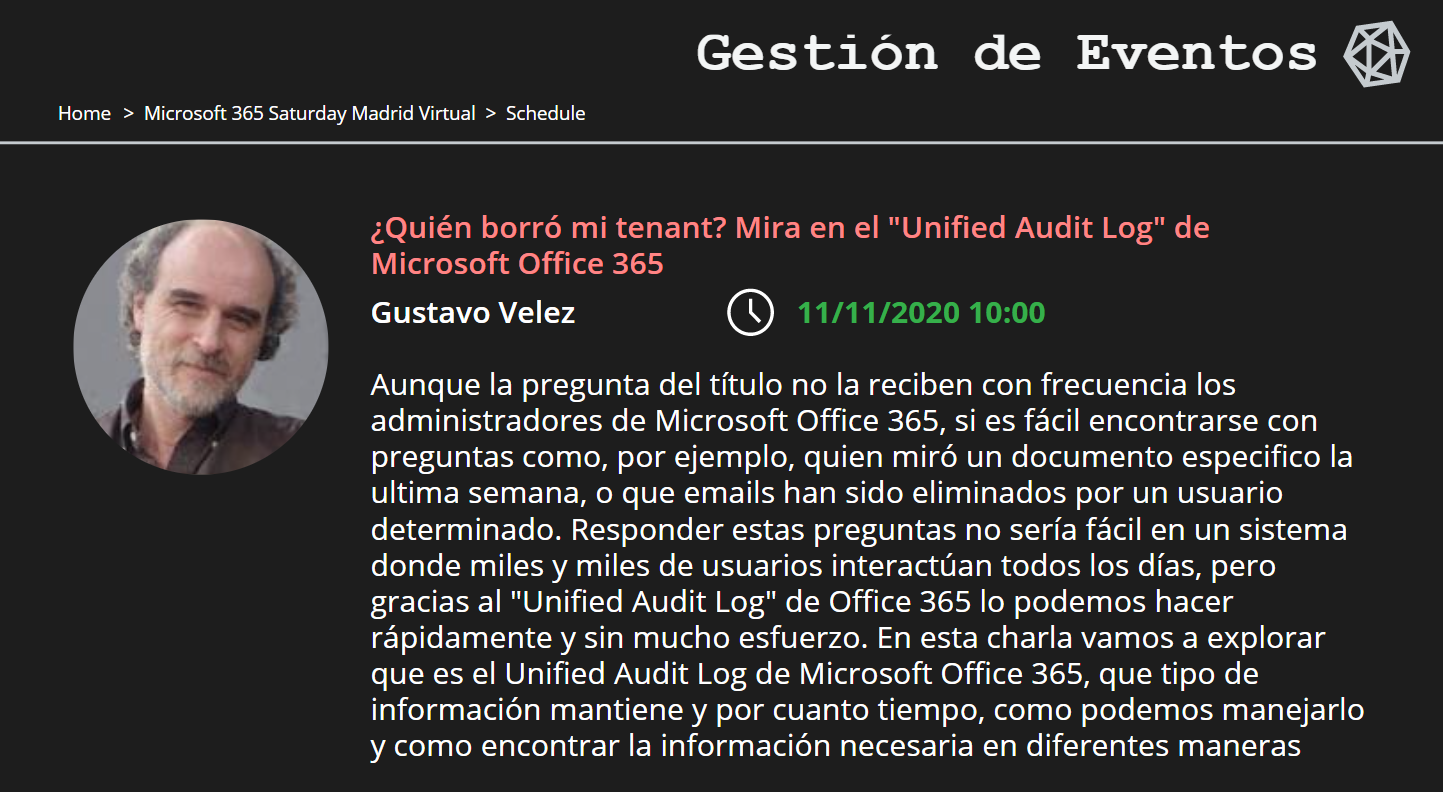
* Localizar las aplicaciones en una home del sitio



* Poder visualizar por cada evento las sesiones, los speaker y el horario de las mismas



* Poder ver un detalle de las sesiones



Durante el laboratorio tendremos la oportunidad de crear la vista de detalle de una sesión, el menú global de navegación y una vista de la agenda, todo esto desarrollado por componentes de PowerApps

Se proporciona junto con este manual un fichero Zip con la solución base que debe incluir:

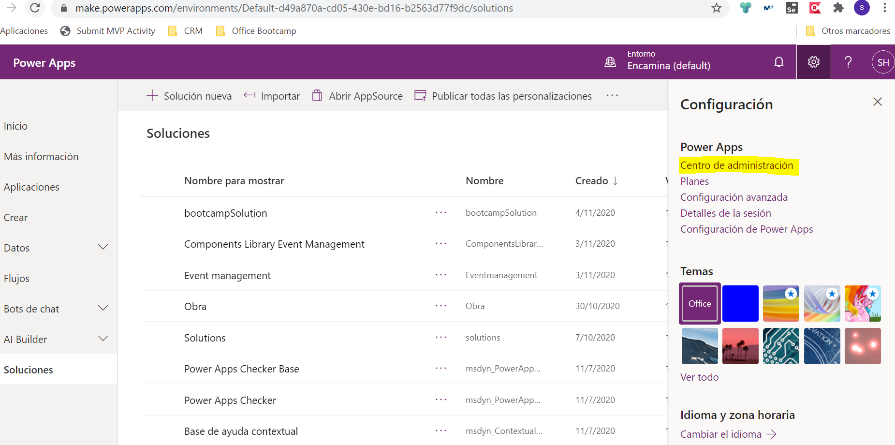
* La app a evolucionar
* Las entidades de CDS:
  + Event: Datos del evento (Nombre, fecha, descripción….)
  + Session: Dato de la sesión (Día de programación, speaker, título, descripción…)
  + Schedule: Cada día que dura el evento (Relaciona la sesión con la jornada y el evento)
  + Speaker: Datos del speaker, se relaciona con la sesión

Todos los datos los importaremos desde unos ficheros CSV para cada una de las entidades, que también son proporcionados junto con este manual.

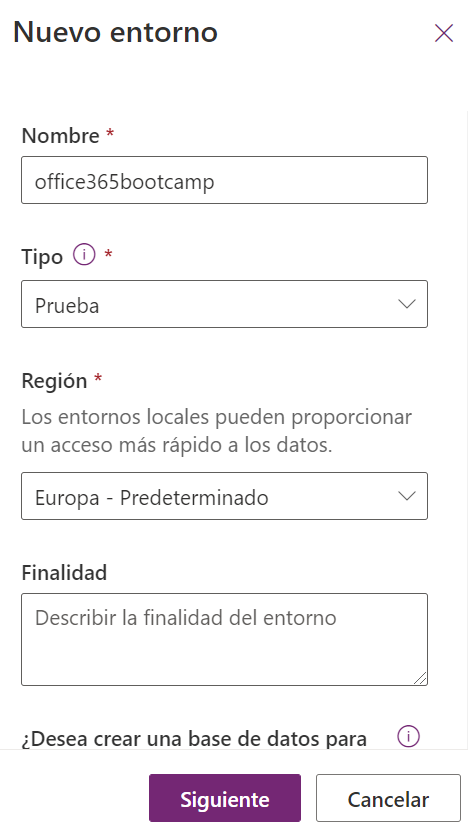
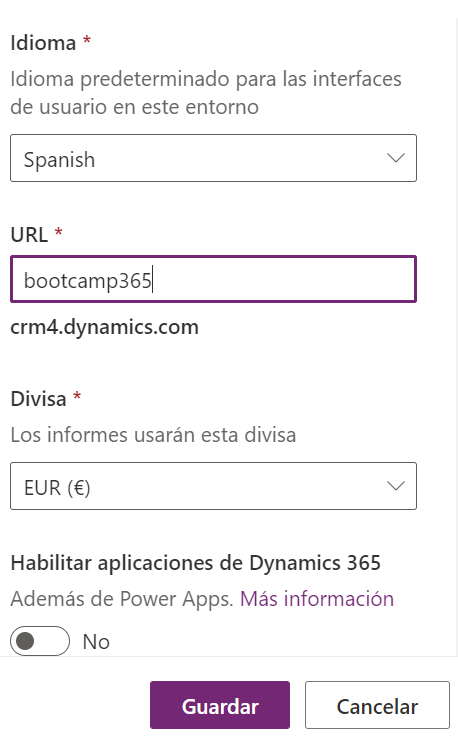
# **Ejercicio 1: Desplegar la solución base en nuestro entorno**

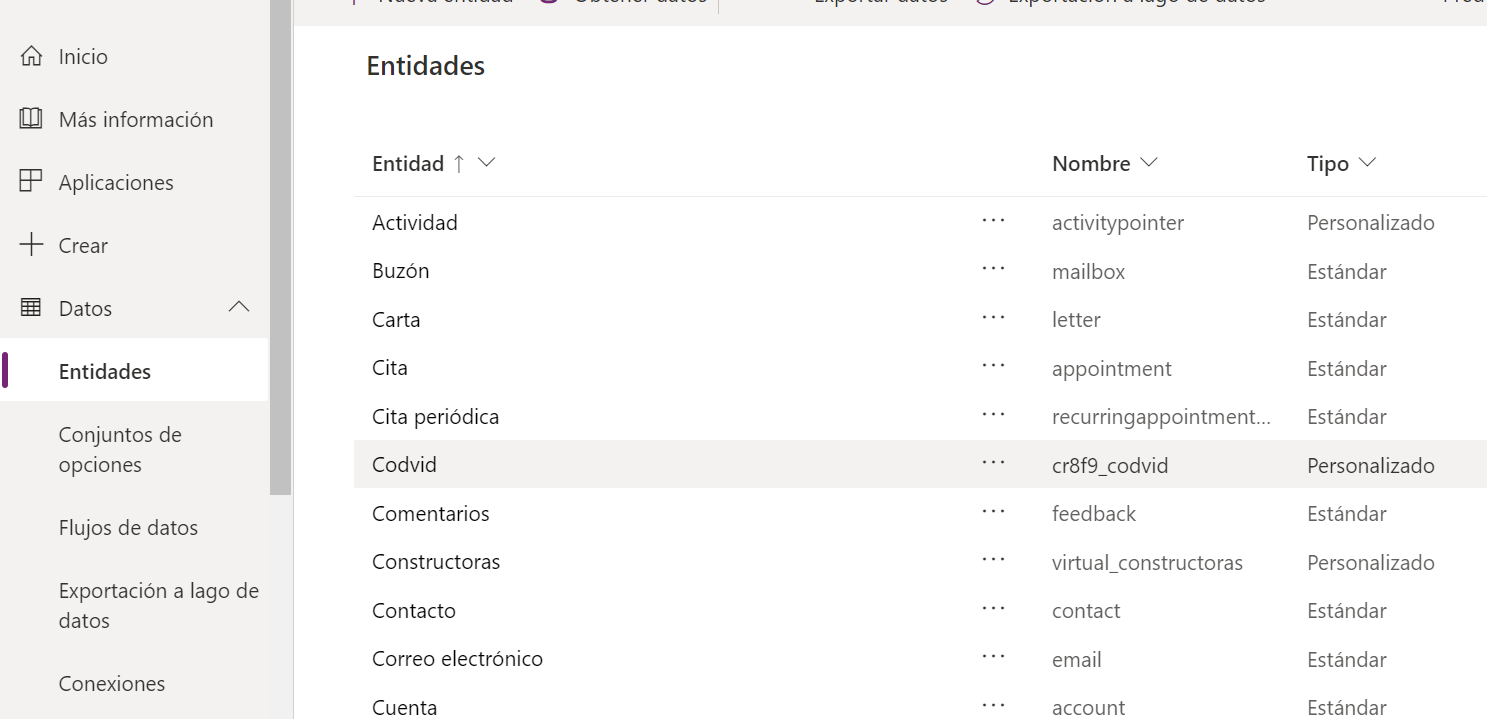
Este primer ejercicio nos va a servir para poner en practica la gestión de entornos y soluciones en Power Apps.

Una vez tengamos nuestro tenant aprovisionado, deberemos:

1. Crear un entorno de Test (podemos usar el default si nos hemos creado un trial antes de empezar el laboratorio).  
     
   Para ello vamos acceder a “Centro de administración” en <http://make.powerapps.com>, con usuario con licencia de Power Apps.

Una vez en el centro de administración, en “Entornos”, seleccionamos en nuevo, y escogemos un test de tipo “Prueba”. Este entorno tendrá un CDS que no impactará en nuestra capacidad del tenant, y se borrará a los 30 días de su creación.

1. Una vez creado el entorno que puede tardar unos minutos, deberemos comprobar que tenemos CDS al entorno, si no nos mostrará un mensaje para crear una nueva base de datos. Esto lo podemos comprobar en la web de Power Apps, en el apartado Datos/Entidades, y deberíamos ver algo parecido a lo de la siguiente imagen.  
     
   

Una vez comprobado que tenemos CDS instanciado en el entorno, vamos a cargar la solución. Para ello accedemos en el menú lateral izquierdo en el menú de “Soluciones”, y seleccionamos en **“Importar”.**

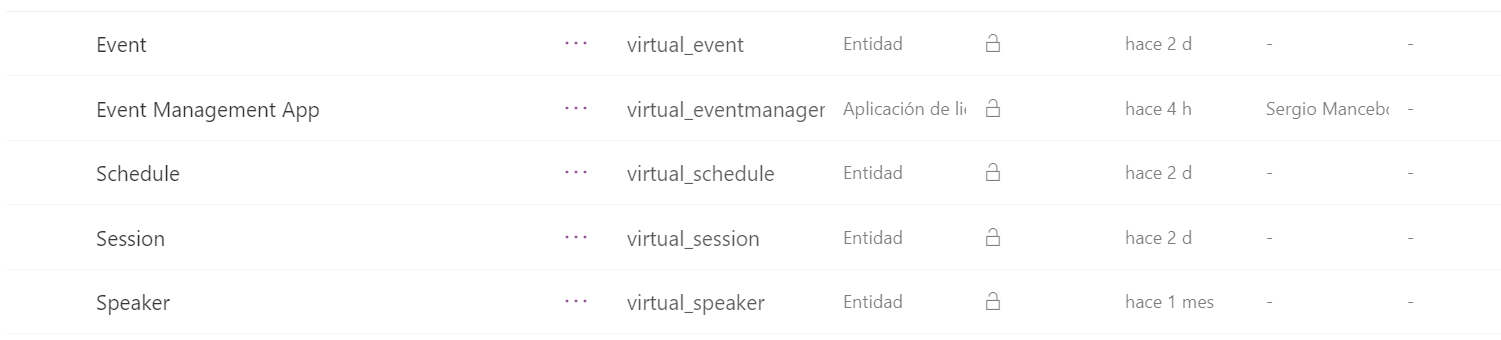
Deberemos subir el zip “**Eventmanagement\_managed.zip**” que tenemos con los datos del laboratorio.



Si el paquete es correcto, el primer paso se nos ofrecerán los detalles del mismo en pantalla, veremos que es una solución de tipo administrada (no le podremos añadir más elementos en nuestro entorno), y deberemos confirmar con “Importar”.

De nuevo el proceso de importación puede durar unos minutos.

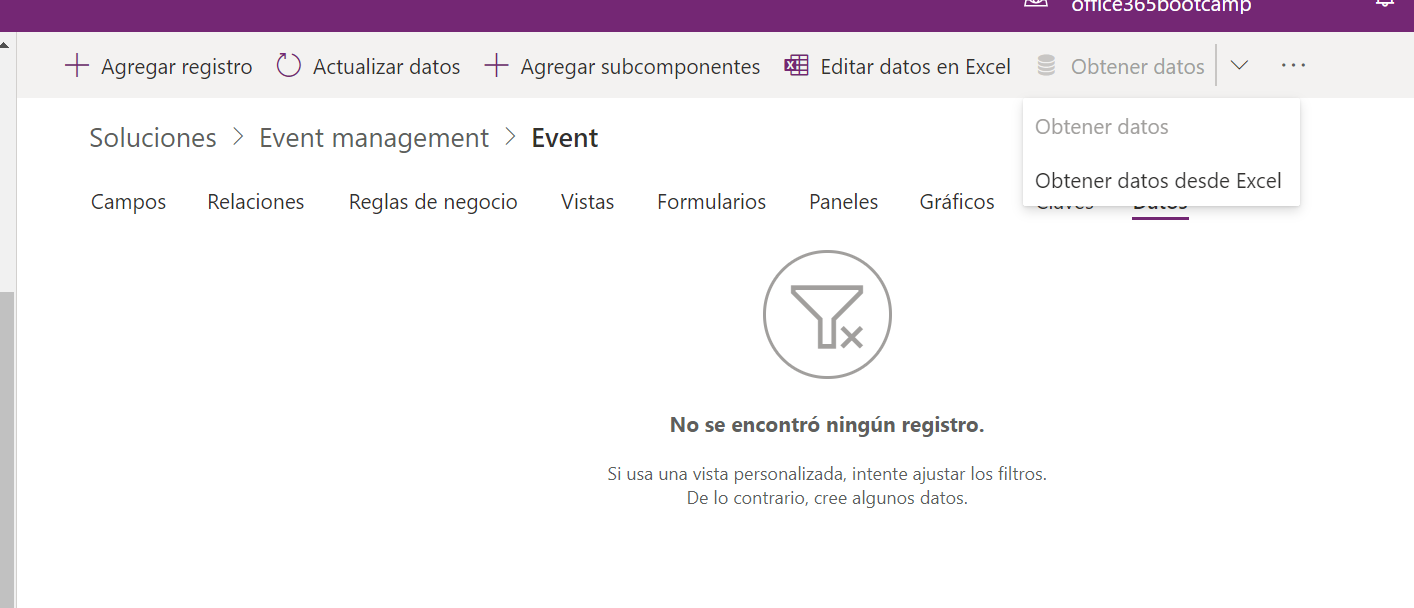
1. Una vez cargada la solución, podremos ver el contenido de la misma, seleccionando en Soluciones/Event Management.

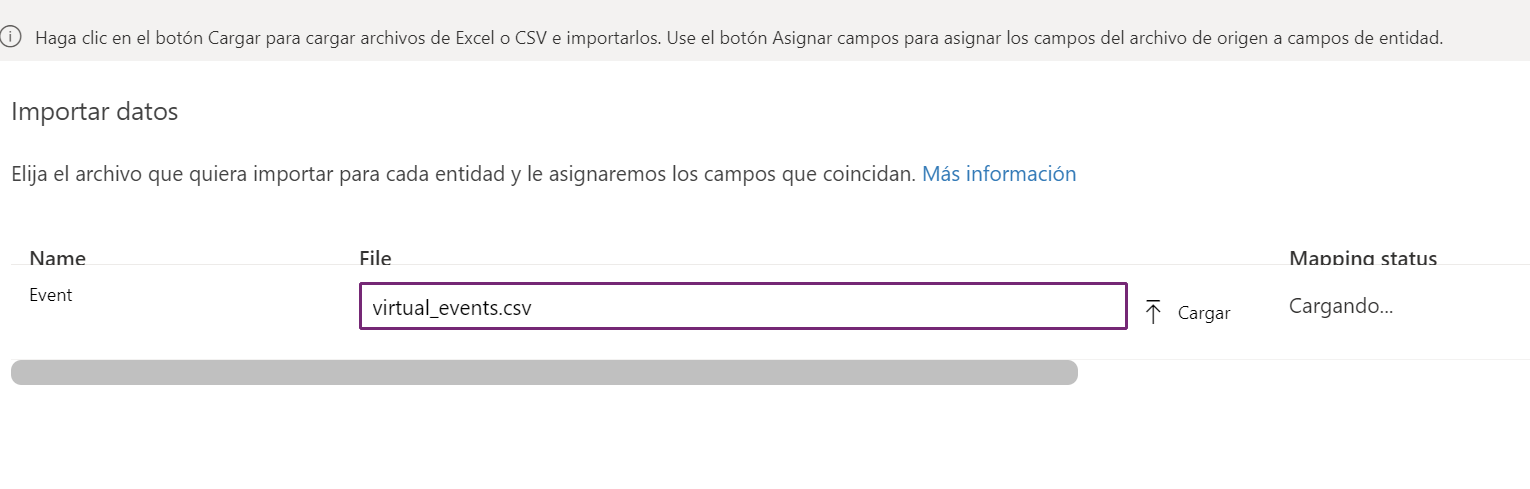


Si todo ha ido bien, podremos ver las **4 entidades** que necesitamos en CDS, y la aplicación de lienzo.

1. Se han cargado las entidades, pero es necesario, cargar datos en ella porque vendrá solo el schema de las mismas, relaciones y formularios, pero no datos pre-cargados.

Para ello vamos a entrar en cada una de las entidades y vamos a cargar los datos de la siguiente forma:

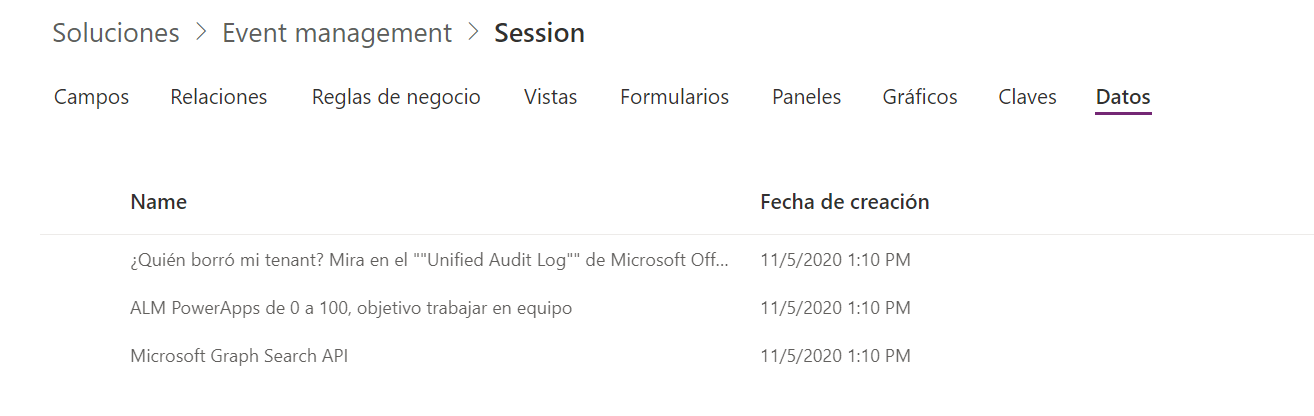
* 1. Abrir el Zip “Exported\_Files”, que contiene un CSV por cada entidad.
  2. Acceder a la entidad que queremos cargar desde Soluciones/Event Management/”Entidad”
  3. En la pestaña de datos de la Entidad, en el menú superior seleccionamos “Obtener datos desde Excel”  
       
     
  4. Deberemos cargar el CSV correspondiente para cada entidad, por lo que deberemos exportar a una carpeta en nuestro local todos los CSV, para poder subirlo para cada entidad.



Deberemos seleccionar “Asignar campos”, revisar que no hay ningún conflicto de mapeo, y confirmar en la parte superior derecha en “Importar”.

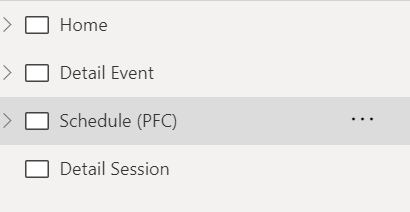
Debemos seguir el siguiente orden de inserción, para no tener conflictos:

* Event
* Speaker
* Schedule
* Session
  1. Podremos comprobar para cada entidad que carguemos, accediendo a la pestaña de datos, que ahora si tenemos datos cargados.



* 1. Por último, vamos a ver que la aplicación funciona de forma correcta, accediendo a la aplicación de lienzo desde la solución.  
       
     En la solución Event management, seleccionamos en “Event Management App”, y en luego en “Editar”.

La aplicación tiene que tener el siguiente árbol de navegación:



**Home**

Pantalla con una galería, que va a listar el contenido de la Entidad “Event”.

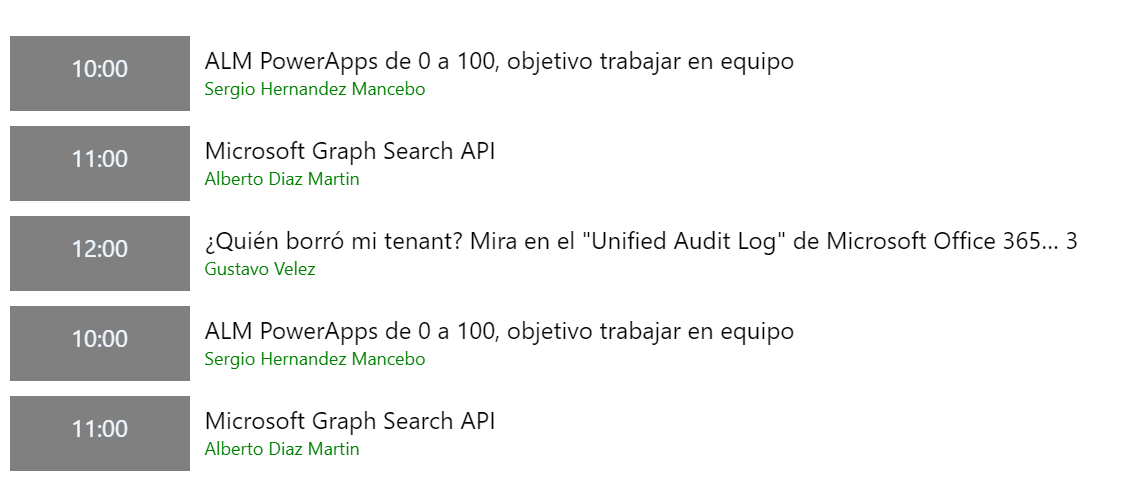


**Detail Event**

Pantalla vinculada al elemento “seleccionado” de Home, y que pinta la sesión en pantalla.  
Además se añade un enlace a la pantalla “Schedule”.  
  


**Schedule**

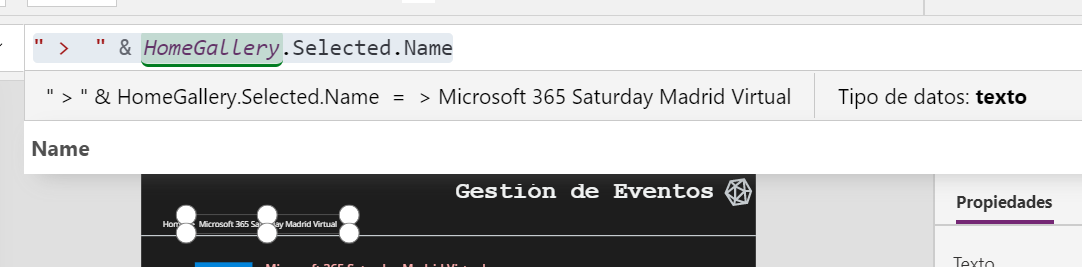
Listado de sesiones del evento, con dos formas de visualizarlas:

* Carrusel o galería de sesiones con la foto del Speaker  
  
* Agenda (Pendiente de desarrollo) : Mostrará la agenda y el horario de las sesiones  
    
  
* Detalle de sesión (Pendiente de desarrollo): Deberá mostrar el detalle de una sesión seleccionada desde la galería de sesiones en la vista “Sessions”.Podemos cerrar este punto, probando la aplicación, y viendoque todo funciona de forma correcta, antes de empezar a evolucionarla.

# **Ejercicio 2: Crear un menú global y una pagina de detalle de sesión con Bibliotecas de componentes**

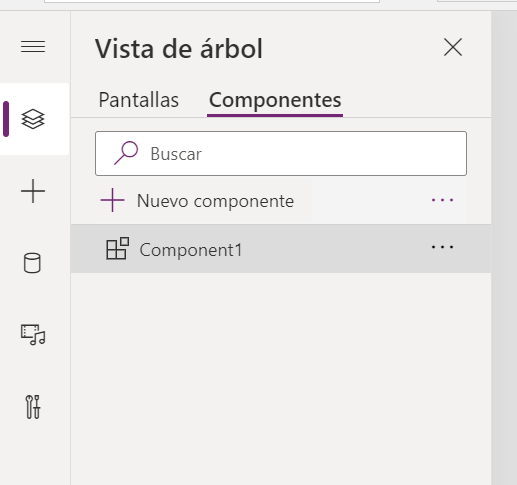
### **CREACIÓN DE UN MENU CÓMUN CON COMPONENTES**

Si analizamos la aplicación al detalle, veremos que el menú de navegación con la el “breadcrumb”, es un copiar pegar de labels y formulas de una página a otra.  
Ahora mismo si accedemos a la página por ejemplo de “Detail Event”, y seleccionamos sobre el menú veremos que tenemos dos labels:

* **Home – Label**: Que tiene un texto fijo de “Home”, y en el evento OnClick tiene un navegar a la Home.
* El **breadCrum**, que tiene un texto de la forma  **" > " & HomeGallery.Selected.Name”,** que pinta el nombre de la sesión seleccionada en la HomeGallery, y un navegar siempre a la Home, que es la página vinculada.  
    
  

Para evitar tener que pegar y copiar por todas las pantallas este menú, vamos a construir un componente de la siguiente forma:

1. Accedemos al menú de Aplicaciones en el menú lateral derecho, y seleccionamos en   
   “**Bibliotecas de componentes – versión preliminar**”, creando una nueva biblioteca, que llamaremos “**Event Management Library**”.  
     
   Esto nos abrirá un Power Apps Studio, que contendrá en la vista de árbol un menú de pantallas, y un menú de componentes.



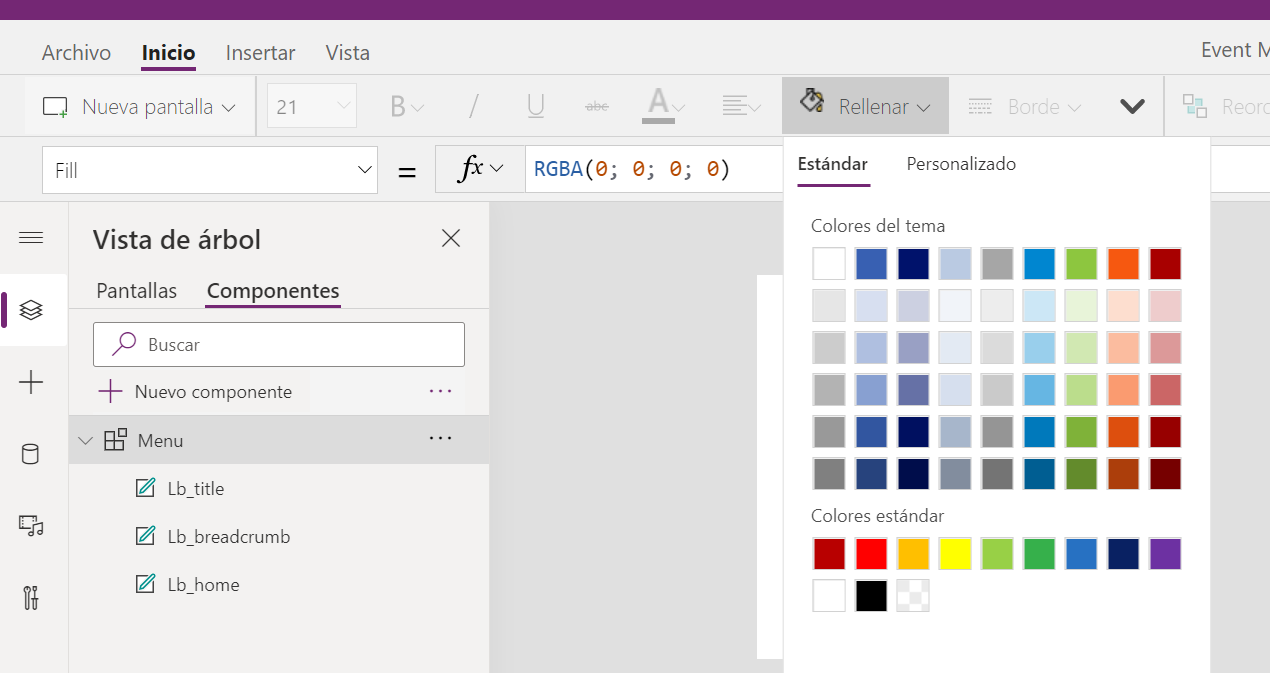
1. Vamos a renombrar el componente base como “Menu”.
2. Vamos desde el menú superior insertar, a insertar 3 Label que llamaremos:  
   * **Lb\_home**
   * **Lb\_ breadcrumb**
   * **Lb\_title**
3. Vamos a darle un fondo de color negro al componente, seleccionando sobre el componente “Menu” en el árbol de navegación, y en Inicio rellenamos de color negro.  
     
   
4. Vamos a seleccionando el componente menú, en el detalle del componente a la derecha del Power Apps Studio, asignarle un **Ancho de 1366px y un Alto de 200 px.**
5. Ahora nos quedaría dar a cada label una fuente de color blanco en el menú de inicio, seleccionando todos los label, y luego colocaremos en pantalla siguiendo un diseño similar al de la app de ejemplo que hemos importado anteriormente.



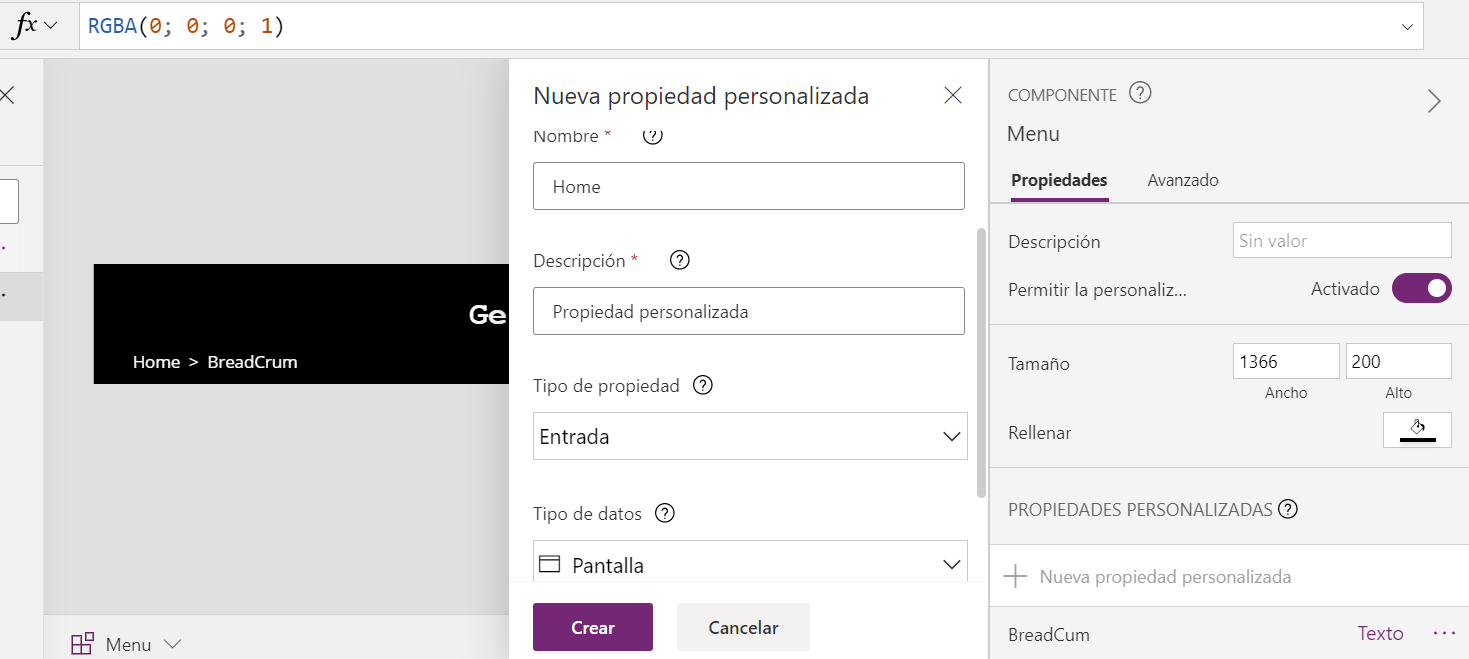
Ilustración Diseño original menú

1. Deberemos ponerle a Lb\_Title en el valor **Text** en la barra de fórmulas, el texto “Gestión de Eventos”, así como una **fuente “Courier New”, un tamaño de fuente de 40** y **SemiBold**.
2. A **Lb\_home** le daremos el valor de texto “**Home**”, y al **Lb\_breadcrum** por ahora le pondremos en la barra de formulas par el texto la siguiente:

**"> " & " BreadCrum"**

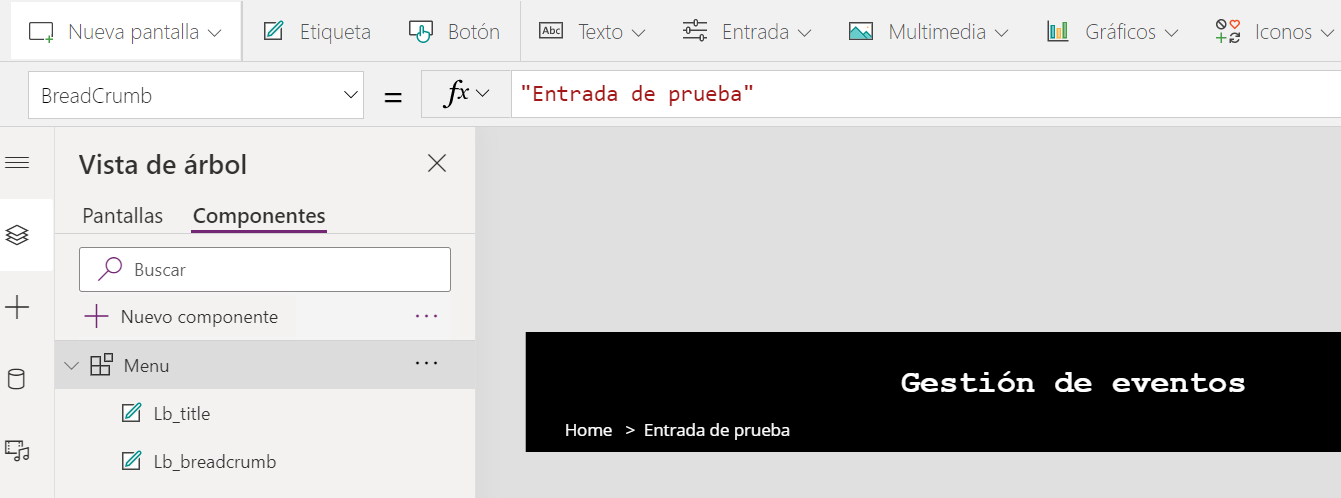
Si vemos el diseño se debe ir pareciendo un poco al de la siguiente imagen  
  
  
  
Lo podríamos dejar así, o añadir los iconos que necesitemos, en el mi caso si vemos el diseño original se añadió un icono junto al texto, y una barra justo debajo de los label gris, que podeis encontrar en el desplegable Iconos en el menú de Insertar.

1. Deberemos seleccionar el componente menú, para poder configurar las entradas y las salidas. Para esto deberemos en el menú de “Propiedades Personalizadas” del componente, seleccionar en “**Nueva Propiedad personalizada**”.  
     
   Deberemos añadir las siguientes propiedades:  
   * **BreadCrumb**: Del tipo entrada y con tipo de datos **Texto**
   * **Navigation**: Del tipo entrada y con el tipo de datos **Pantalla**
   * **Home**: Del tipo entrada y con el tipo de datos **Pantalla**



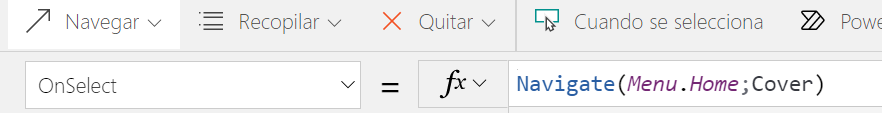
1. Deberemos editar las propiedades Text de Lb\_breadcrumb con el siguiente valor

***If(Menu.BreadCrumb = "";"";Text( " > " & Menu.BreadCrumb;"[$-es-ES]"))***

Con esto conseguiremos pintar en el texto, la miga de pan de forma dinámica con el parámetro de entrada al componente breadCrumb.  
  
Podremos probarlo si seleccionamos sobre el componente, y en la propiedad BreadCrumb le asignamos por ejemplo el valor “Entrada de prueba”.  
  


1. Para terminar el menú, nos queda vincular las propiedades de tipo pantalla de inicio, con los eventos OnSelect de los labels Home y BreadCrum:  
     
   **BreadCrumb: Navigate(Menu.Navigation;ScreenTransition.Cover)**

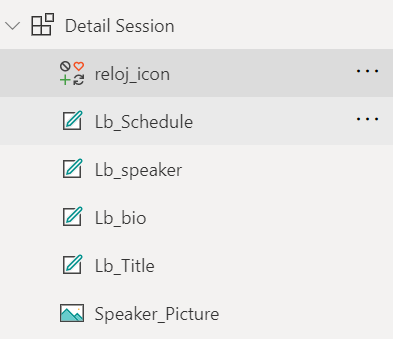
**Home: Navigate(Menu.Home;ScreenTransition.Cover)**



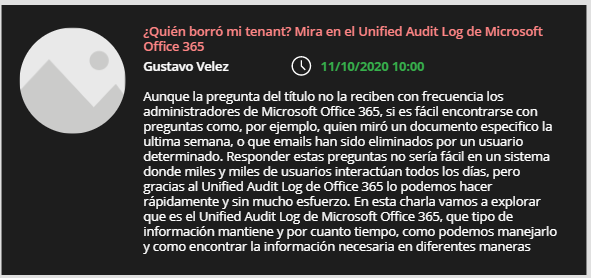
### **CREACIÓN DE PAGINA DE DETALLE SESION CON COMPONENTES**

Una vez tenemos el menú, vamos a crear el componente “Detail Session”, pulsando en “Agregar nuevo componente” en la vista de árbol.

Vamos a dar por hecho que tenemos una visión mínima de Power Apps, y queremos recrear el siguiente árbol de navegación:



La pantalla que queremos diseñar debe tener la siguiente pinta, o podemos hacerlo un poco estilo libre:



Como vemos en la imagen, vamos a tener una imagen a la izquierda para la foto del Speaker, un título de la sesión en rojo, y justo de bajo el nombre del Speaker, la hora de la sesión (en verde y con un logo de reloj), y por último añadimos el label bio.

El tamaño del competente debe de ser de **1300 de Ancho y 600 de Alto**

Lo interesante nuevamente aquí, es definir de forma correcta las propiedades de entrada y de salida del componente:

|  |  |  |
| --- | --- | --- |
| Propiedad | Entrada / Salida | Tipo |
| Picture | Entrada | Imagen |
| Speaker | Entrada | Texto |
| Title | Entrada | Texto |
| Schedule | Entrada | Texto |
| Bio | Entrada | Texto |

Ahora tenemos que vincular cada entrada de texto con su label correspondiente, tal y como hicimos en el ejemplo del menú:

|  |  |
| --- | --- |
| Label | Formula para Text |
| Lb\_speaker | 'Detail Session'.Speaker |
| Lb\_title | 'Detail Session'.Title |
| Lb\_schedule | 'Detail Session'.Schedule |
| Lb\_bio | 'Detail Session'.Bio |

Además deberemos añadirle a la imagen Speaker\_Picture el valor “Detail Session'.Picture”.

Para dar un formato redondeado a la imagen deberemos configurar las siguientes propiedades:

|  |  |
| --- | --- |
| Propiedad | Valor |
| Height | 236 |
| Width | 236 |
| RadiusBottomLeft | Speaker\_Picture.Height |
| RadiusBottomRight | Speaker\_Picture.Height |
| RadiusTopLeft | Speaker\_Picture.Height |
| RadiusTopRight | Speaker\_Picture.Height |

Podremos añadir los siguientes textos por defecto a las propiedades de entrada para poder probar nuestro diseño:

|  |  |
| --- | --- |
| Propiedad | Texto |
| Speaker | Gustavo Velez |
| Title | ¿Quién borró mi tenant? Mira en el Unified Audit Log de Microsoft Office 365" |
| Schedule | 11/10/2020 10:00 |
| Bio | Aunque la pregunta del título no la reciben con frecuencia los administradores de Microsoft Office 365, si es fácil encontrarse con preguntas como, por ejemplo, quien miró un documento especifico la ultima semana, o que emails han sido eliminados por un usuario determinado. Responder estas preguntas no sería fácil en un sistema donde miles y miles de usuarios interactúan todos los días, pero gracias al Unified Audit Log de Office 365 lo podemos hacer rápidamente y sin mucho esfuerzo. En esta charla vamos a explorar que es el Unified Audit Log de Microsoft Office 365, que tipo de información mantiene y por cuanto tiempo, como podemos manejarlo y como encontrar la información necesaria en diferentes maneras |

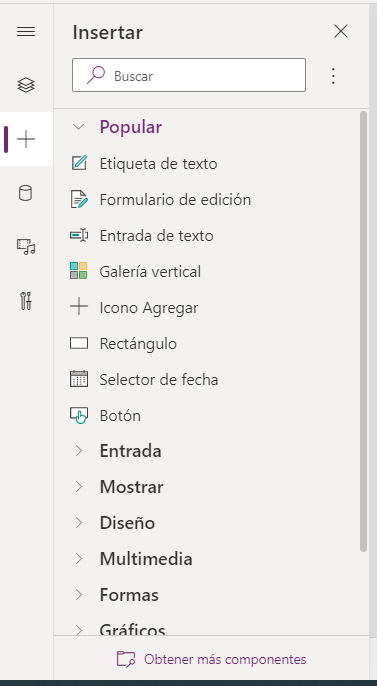
Poniendo valores a las propiedades, podemos ir probando el diseño de la pantalla.

Deberemos en este momento, guardar la biblioteca de componentes, y publicar la misma.

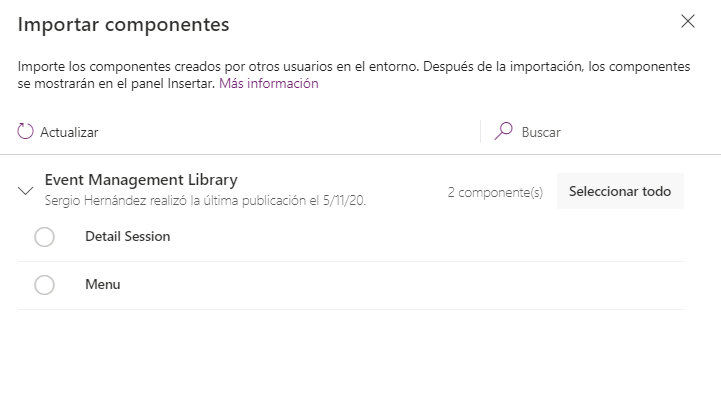
### **INTEGRAR COMPONENTES EN EL APP PRINCIPAL**

Debemos acceder a la aplicación importada Event Management App para poder empezar a usar el Componente de Menu, y el Componente de Detalle de una sesión.

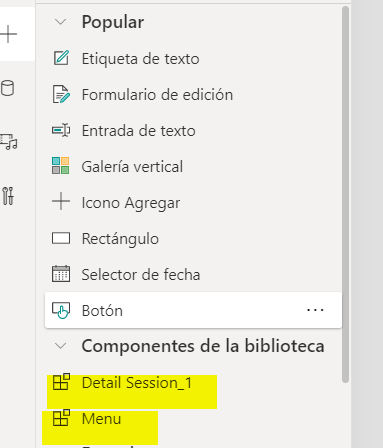
Para ello desde la aplicación, en el menú lateral izquierdo seleccionamos Insertar, y justo debajo del árbol de navegación, se nos presenta un enlace con el texto “**Obtener más componentes**”.



Seleccionamos en el enlace, y en la parte derecha, veremos que aparece los dos componentes de “Event Management Library”, que hemos creado antes:



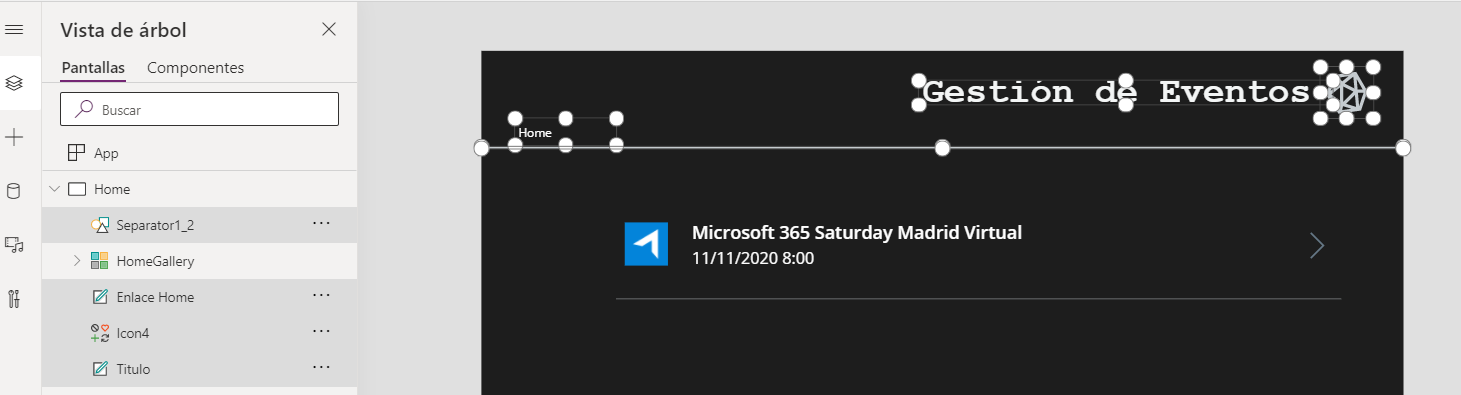
Los seleccionamos y seleccionamos “Importar”, para que nos aparezcan los componentes en la biblioteca de componentes de la aplicación.



Desde este momento, podemos insertar los componentes en la aplicación, como cualquier otro elemento en una Power Apps.

### Integrar el componente Menu

Para probar el menú, deberemos borrar todos los elementos del menú, que tenemos en la página de Home, por ejemplo, e insertar el Componente Home en la página renombrándolo a Menu.

Como veis en la imagen, los datos del menú siguen siendo de prueba, ya que no he cambiado las propiedades de entrada y salida.

Deberemos hacer esto en todas las pantallas teniendo unas propiedades de entrada según muestra la siguiente tabla, teniendo en cuenta que BreadCrumb tendrá un valor texto, y Navigation y Home siempre un valor pantalla para poder simular la navegación:

|  |  |  |
| --- | --- | --- |
| Pantalla | Propiedad | Valor |
| Home | BreadCrumb | Sin valor |
|  | Navigation | Sin valor |
|  | Home | Home |
| Detail Event | BreadCrumb | HomeGallery.Selected.Name |
|  | Navigation | Home |
|  | Home | Home |
| Schedule PCF | BreadCrumb | HomeGallery.Selected.Name |
|  | Navigation | Detail Event |
|  | Home | Home |
| Detail Session | BreadCrumb | SessionGallery.Selected.Name |
|  | Navigation | Detail Event |
|  | Home | Home |

Si no nos hemos equivocado en alguna propiedad, o al poner el nombre de las pantallas de otra forma (cosa que podemos corregir sobre la marcha), deberíamos tener todas las pantallas enlazadas por el menú, dese un componente centralizado.

### **Intregrar el componente “Detail Session”**

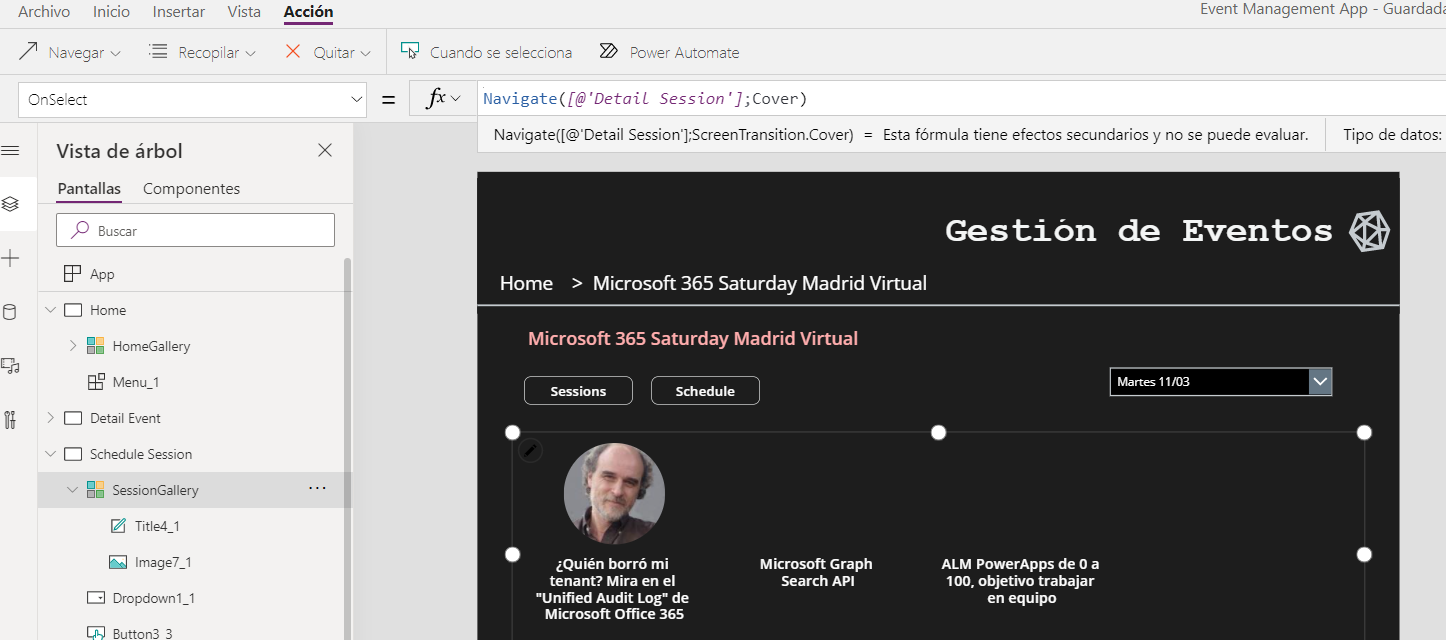
Para integrar la pantalla de detalle de la sesión, haremos un proceso muy similar al Menú. Accedemos a la página “Detail Session” que ahora mismo solo debería tener un menú, y le insertamos el componente de detalle.

Deberemos configurar el componente con las siguiente propiedades:

|  |  |
| --- | --- |
| Propiedad | Valor |
| Picture | SessionGallery.Selected.'Speaker Session'.Picture |
| Speaker | SessionGallery.Selected.'Speaker Session'.UserName |
| Title | SessionGallery.Selected.Name |
| Schedule | SessionGallery.Selected.Time |
| Bio | SessionGallery.Selected.Bio |

Por último nos quedaría vincular la página “Schedule PCF con la nueva pantalla de detalle de una sesión”.

Para conseguirlo, seleccionamos en SessionGallery, y en Acciones le configuramos el evento OnSelect con el valor **Navigate([@'Detail Session'];ScreenTransition.Cover)**

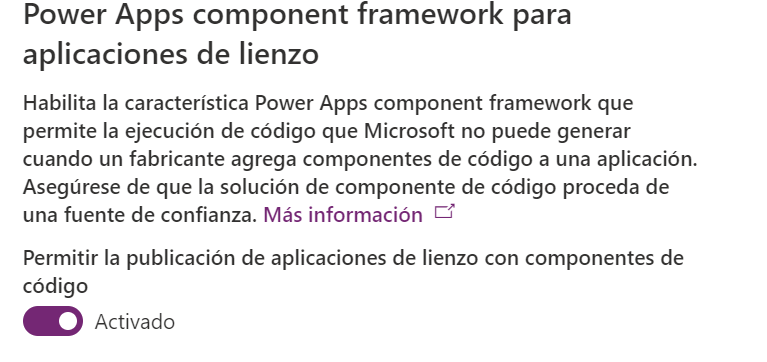


Podemos probar toda la app, ya que una vez añadido este último evento tenemos 100% integrados el menú y la vista de detalle.

# **Ejercicio 3: Crear un componente con Power App Component Framework en React con VS Code**

Es importante tener en cuenta que para que podamos hacer uso de “Componentes debemos antes de nada hacer lo siguiente:

* 1. Acceder a la gestión de entorno, en el centro de administración.
  2. Seleccionar configurar el entorno que estamos utilizando, donde hemos ido haciendo este laboratorio.
  3. En el apartado de producto seleccionamos en Características
  4. Activamos la opción: Power Apps component framework para aplicaciones de lienzo.



### Preparar la aplicación

Vamos a preparar la aplicación para añadir el componente “Agenda” que vamos a desarrollar en React con el framework de Componentes de Power Apps.

Vamos a añadir a la pagina actual una lógica para poder ocultar la agenda, o el listado de sesiones, siguiendo los siguientes pasos en la página de Schedule (PCF)

1. Sobre la interfaz seleccionamos los botones de “Session” y “Schedule” y los renombramos a “btnSession” y “btnSchedule”



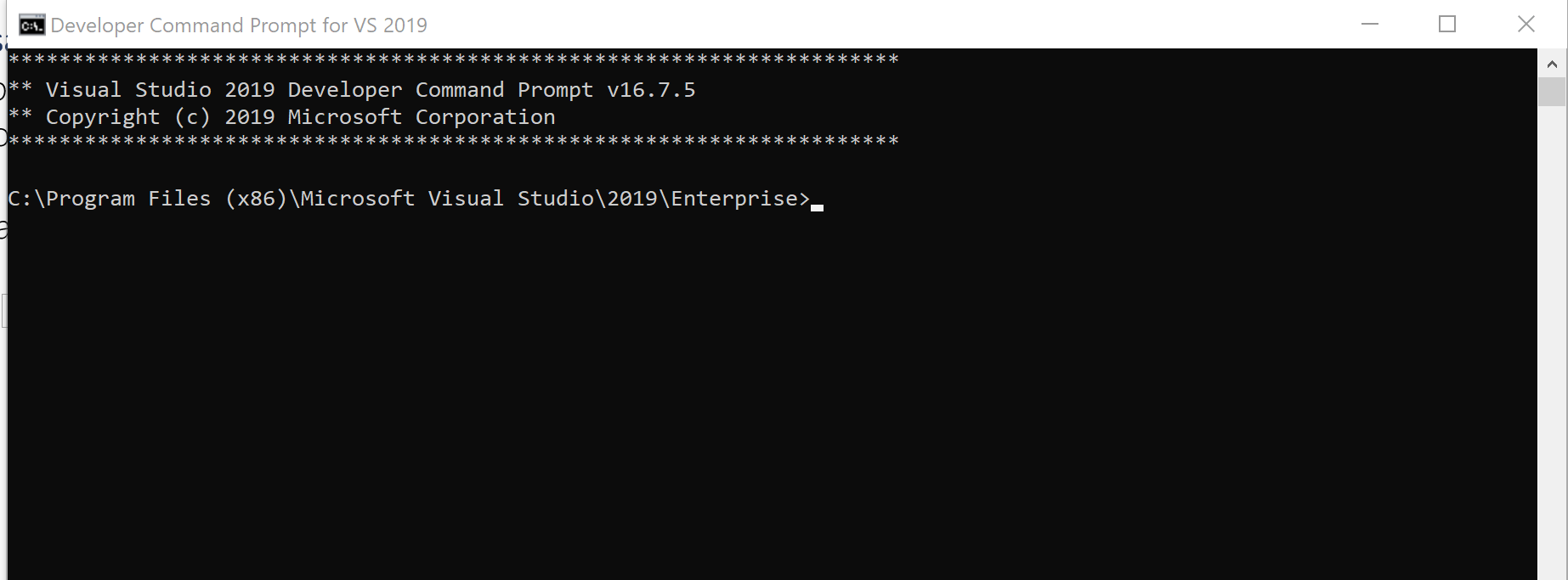
1. Creamos una pagina nueva copiando la de Schedule y la vamos a pasar a llamar Session Gallery.
2. Borramos de Schedule la galería de sesiones, y en la nueva pagina renombramos la galería a Session Gallery.
3. Le añadimos la siguiente formulas a estos dos botones en ambas pantallas:

|  |  |  |
| --- | --- | --- |
| Boton | Propiedad | Formula |
| btnSchedule | OnSelect | Navigate([@'Schedule (PCF)']) |
| btnSession | OnSelect | Navigate([@'Session Gallery']) |

### **Desarrollar el componente Agenda con Power App Component Framework**

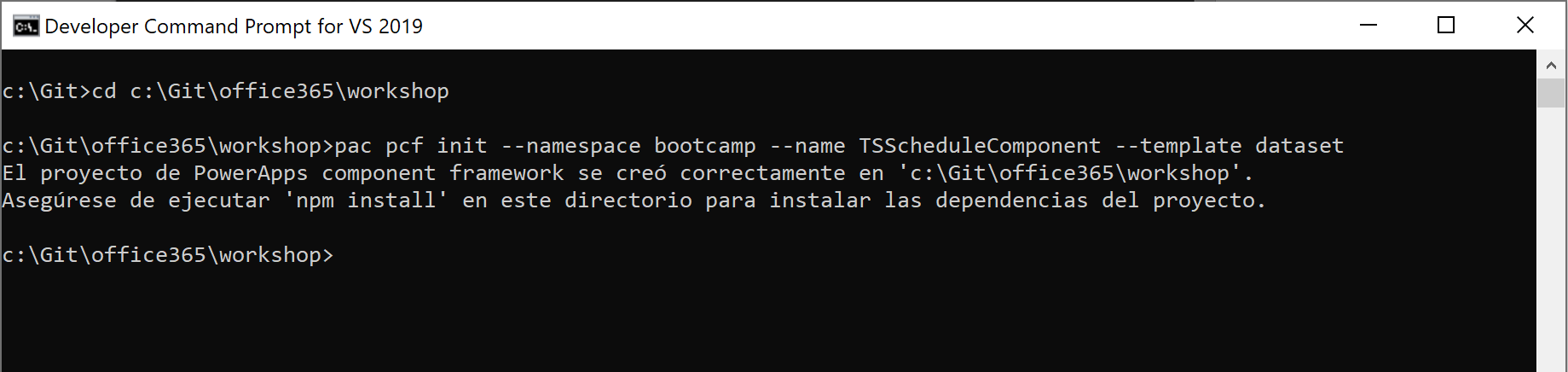
Debemos de asegurarnos de haber instalado correctamente todos los pre-requisitos del inicio del laboratorio para poder desarrollar componente de código.

Para desarrollar el componente de Agenda, vamos a seguir los siguientes pasos:

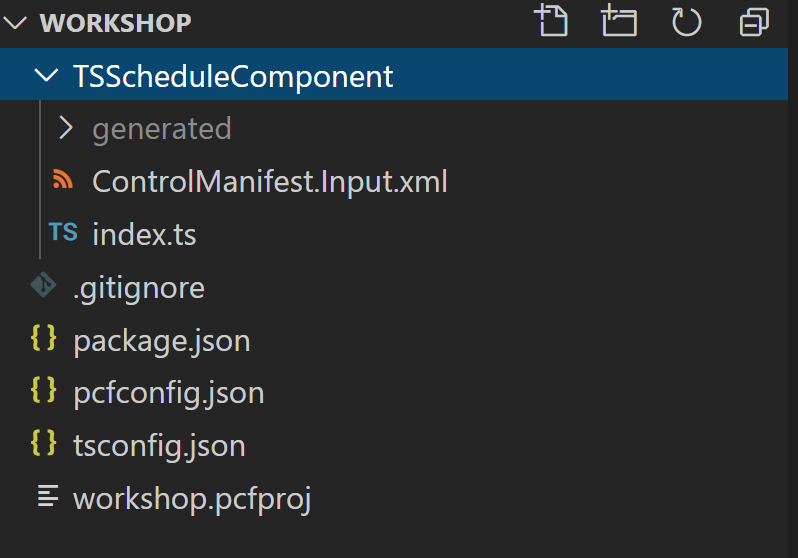
1. Abrimos la consola de Developer Command Prompt VS2019 o superior.  
   
2. Elegimos una ruta donde desarrollar el componente, en mi caso C:\Git\o365\workshop, y accedemos por consola a ella



1. Ejecutamos el siguiente comando por consola:  
   **pac pcf init --namespace** bootcamp **--name** TSScheduleComponent **--template dataset**



1. Ejecutamos el comando **code .** , para poder acceder desde VS Code al código.
2. Podemos observar que se ha creado una estructura de proyecto como la de la siguiente imagen, y vamos a estar muy atentos al fichero **ControlIManifest y al Idext.ts**



1. Ahora vamos a empezar a configurar nuestro componente, para ello en el manifiesto del componente vamos a sustituirlo por el siguiente código:

<?xml version="1.0" encoding="utf-8" ?>

<manifest>

  <control namespace="bootcamp365" constructor="TSScheduleComponent" version="0.0.1" display-name-key="TSScheduleComponent" description-key="TSScheduleComponent description" control-type="standard">

<!-- dataset node represents a set of entity records on CDS; allow more than one datasets -->

    <data-set name="sampleDataSet" display-name-key="Dataset\_Display\_Key">

      <property-set name="Schedule" display-name-key="Schedule" description-key="Schedule of the session" of-type="SingleLine.Text" usage="bound" required="true" />

      <property-set name="Session" display-name-key="Session" description-key="Session name" of-type="SingleLine.Text" usage="bound" required="true" />

      <property-set name="Speaker" display-name-key="Speaker" description-key="Speaker of the session" of-type="SingleLine.Text" usage="bound" required="true" />

    </data-set>

    <resources>

      <code path="index.ts" order="1"/>

      <css path="Styles/styles.css" order="1" />

    </resources>

  </control>

</manifest>

Si analizamos el manifiesto, vemos que hemos añadido una propiedad de entrada del tipo “data set” con el **tag <data set>,** y que esta propiedad tiene una serie de campos que son los **tag <property-set>.**

Esto lo hacemos para que nuestra agenda acepte por entrada una colección de datos o un CSV por ejemplo.

Ahora tenemos que editar el fichero **Index.ts**, que es el encargado de definir nuestro componente, y deberemos añadir lo siguiente:

En la cabecera del index, vamos a modificar los Imports de la siguiente forma:

import React = require("react");

import ReactDOM = require("react-dom");

import { deflateRawSync } from "zlib";

import {IInputs, IOutputs} from "./generated/ManifestTypes";

import { ScheduleComponent } from "./Schedule/Schedule";

import { SessionScheduleProps, SessionScheduleTable } from "./Schedule/Types/SessionsSchedule";

import DataSetInterfaces = ComponentFramework.PropertyHelper.DataSetApi;

type DataSet = ComponentFramework.PropertyTypes.DataSet;

Por ahora no sabemos que son algunos de ellos, pero los iremos creando poco a poco.

Definimos una variable privada dentro del componente, justo encima del Constructor del mismo, para poder definir un “DIV” contenedor.

export class TSScheduleComponent implements ComponentFramework.StandardControl<IInputs, IOutputs> {

    private theContainer: HTMLDivElement;

    constructor()

    {

    }

Despúes vamos a ir modificando los métodos init y updateview que nos proporciona la plantilla:

* **Init**

public init(context: ComponentFramework.Context<IInputs>, notifyOutputChanged: () => void, state: ComponentFramework.Dictionary, container:HTMLDivElement)

    {

        this.theContainer = container;

    }

* updateView

    public updateView(context: ComponentFramework.Context<IInputs>): void

    {

        context.mode.trackContainerResize(true);

        ReactDOM.render(

            React.createElement(

                ScheduleComponent,this.GetData(context)

            ),this.theContainer

        );

    }

Para el que sepa algo de React, ya ha deducido que nos va a hacer falta definir un component “Schedule Component” que es el que me va a permitir renderizar los datos de la agenda.

Antes de ir al componente de React, nos va a hacer falta definir una función en Index, para poder sacar del data-set definido en el manifiesto, los datos necesarios y transformarlos a tipos que entienda mi componente ScheduleComponent.   
Añadimos un método GetData, dentro del component de Index, para poder sacar los datos necesarios.

private GetData(context: ComponentFramework.Context<IInputs>) : SessionScheduleTable

    {

        let array : SessionScheduleProps[] = [];

        let result: SessionScheduleTable = {ScheduleTable : array};

        let ids = context.parameters.sampleDataSet.sortedRecordIds;

        if (ids.length > 0)

        {

        ids.forEach(function(current)

        {

            let currentRecord = context.parameters.sampleDataSet.records[current];

            result.ScheduleTable.push({

                Schedule: currentRecord.getFormattedValue("Schedule"),

                Speaker :currentRecord.getFormattedValue("Speaker"),

                Title : currentRecord.getFormattedValue("Session")

            })

        }

        );

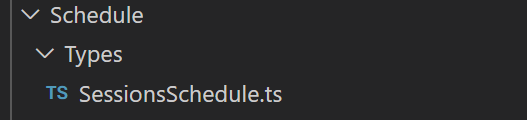
    }

        return result;

}

Ahora necesitamos crearnos una carpeta a nivel de solución dentro del component que vamos a llamar Schedule.

Dentro de Schedule vamos a crearnos una carpeta **Types** en el que crearemos dentro un fichero **SessionSchedule.ts** ,donde vamos a definir unos tipos de datos en typescript que va a utilizar nuestro component de Schedule.



En SessionsSchedule.ts vamos a añadir el siguiente código, para definir los tipos de datos que mapeen nuestro modelo de datos en CDS con el html renderizado por el código del componente.

export type SessionScheduleProps =

{

    Schedule?: string;

    Title?: string;

    Speaker?:string;

}

export type SessionScheduleTable = {

    ScheduleTable:SessionScheduleProps[];

}

Una vez tipado los datos, vamos a crear dentro de la caperta Schedule, un fichero Schedule.tsx, en el que vamos a definir nuestro componente de React.   
Este componente va a recibir la colección de datos de nuestras sesiones, y las va a trasformar en una agenda en pantalla.

El código es el siguiente:

import React = require("react");

import { SessionScheduleProps, SessionScheduleTable } from "./Types/SessionsSchedule";

export class ScheduleComponent extends React.Component<SessionScheduleTable>

{

    constructor(props: SessionScheduleTable) {

        super(props);

    }

// Método para generar el HTML

    getHtml()

    {

        let html : JSX.Element[] = [];

        let index : number = 0;

        this.props.ScheduleTable.forEach(function(item:SessionScheduleProps)

        {

            html.push(

            <div id={"item" + index} className="itemSchedule">

                    <div className="time">{item.Schedule}</div>

                    <div className="sessionData">

                        <div className="title">{item.Title}</div>

                        <div className="speaker">{item.Speaker}</div>

                    </div>

            </div>

            )

            index = index + 1;

        });

        return html;

    }

    public render(): JSX.Element

    {

    return (

            <div id="scheduleContainer">

                {this.getHtml()}

            </div>

        );

    }

}

Si recordáis del manifiesto inicial teníamos una sección “Resources”, donde le indicamos que teníamos un fichero CSS para este componente.

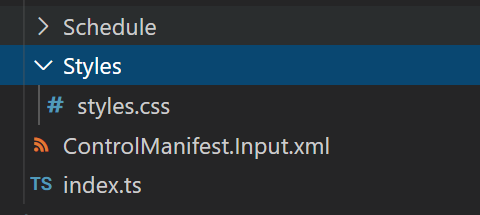
<resources>

      <code path="index.ts" order="1"/>

      <css path="Styles/styles.css" order="1" />

    </resources>

Es importante mantener esa ruta Styles/styles.css, por lo que deberemos crear una carpeta Styles y un fichero styles.css según se ve en la siguiente imagen.



Por último, añadimos el contenido del fichero Css para poder dar estilos a nuestra solución de componente Powerapps.

#scheduleContainer

{

    height: 300px;

    overflow-y: auto;

    overflow-x: hidden;

    width: 850px;

    margin-top: 50px;

    margin-left: 20px;

    background-color: #1D1D1D;

    padding-right: 10px;

    padding-top: 20px;

}

.itemSchedule

{

    height: 40px;

    width: 800px;

    margin-bottom: 40px;

    float: left;

}

.time

{

    float: left;

    background-color: #00B050;

    color: aliceblue;

    margin-right: 5px;

    height: 45px;

    width: 100px;

    text-align: center;

    vertical-align:middle;

    margin-left: 50px;

    padding-top: 5px;

}

.sessionData

{

    width: 600px;

    float: left;

    height: 45px;

    text-align: left;

    background-color: white;

    padding-left: 20px;

    padding-top: 5px;

}

.title

{

    font-size: 16px;

    color: black;

    text-align: left;

    height: 20px;

    margin-bottom: 2px;

}

.speaker

{

    font-size: 12px;

    color: green;

    text-align: left;

    height: 15px;

}

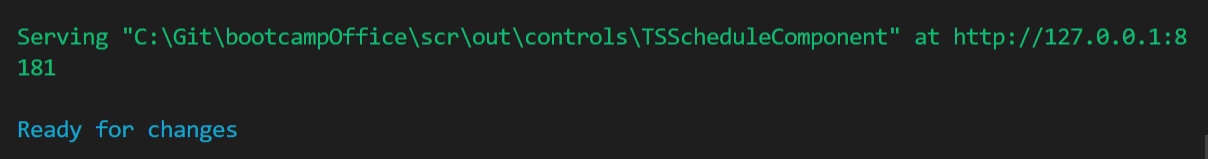
Con este poco código, ya tenemos preparado toda la solución, si todo es correcto deberíamos pasar a compilar nuestro componente y a instalar todas sus dependencias.

Para ello ejecutamos los dos siguientes scripts por la línea de comandos:

* **npm install**
* **npm run build**

### Depurar en local y probar su ejecución

Ya hemos terminado de trabajar en el componente, por lo que podemos recompilar la solución con **“npm run build”,** y posteriormente podemos lanzarla desde el terminal de VS Code con **“npm run start”**

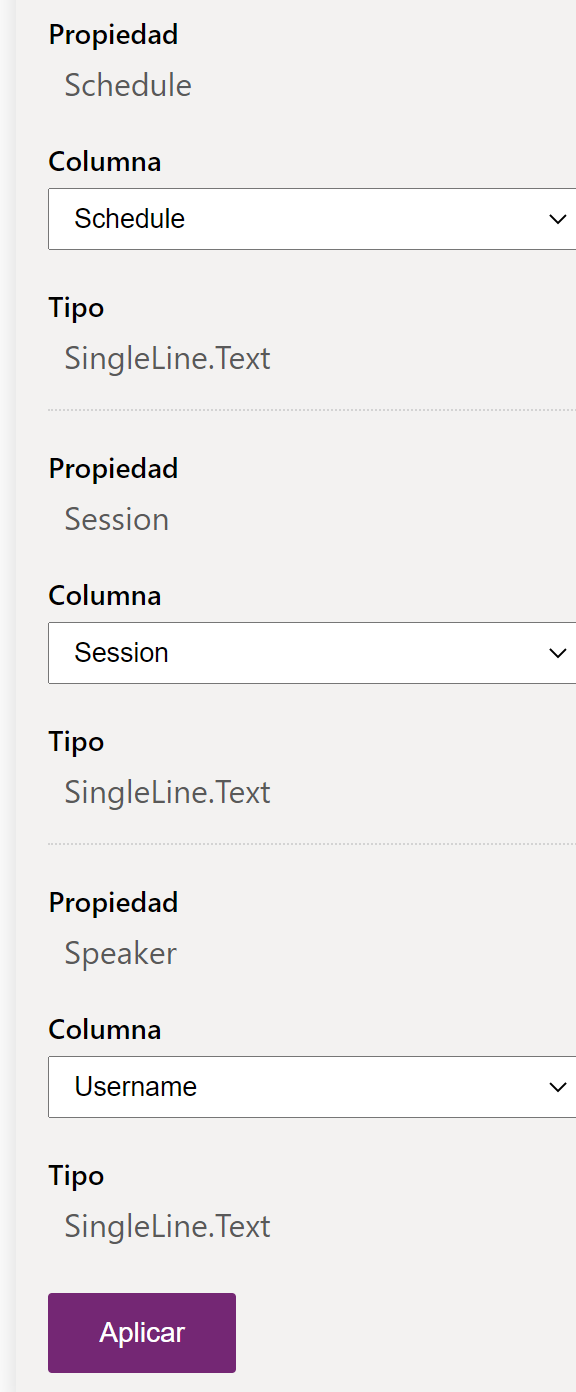


Una vez arrancada la app, podremos visualizarla en el navegador, y tendrá una pinta como la de la siguiente imagen.

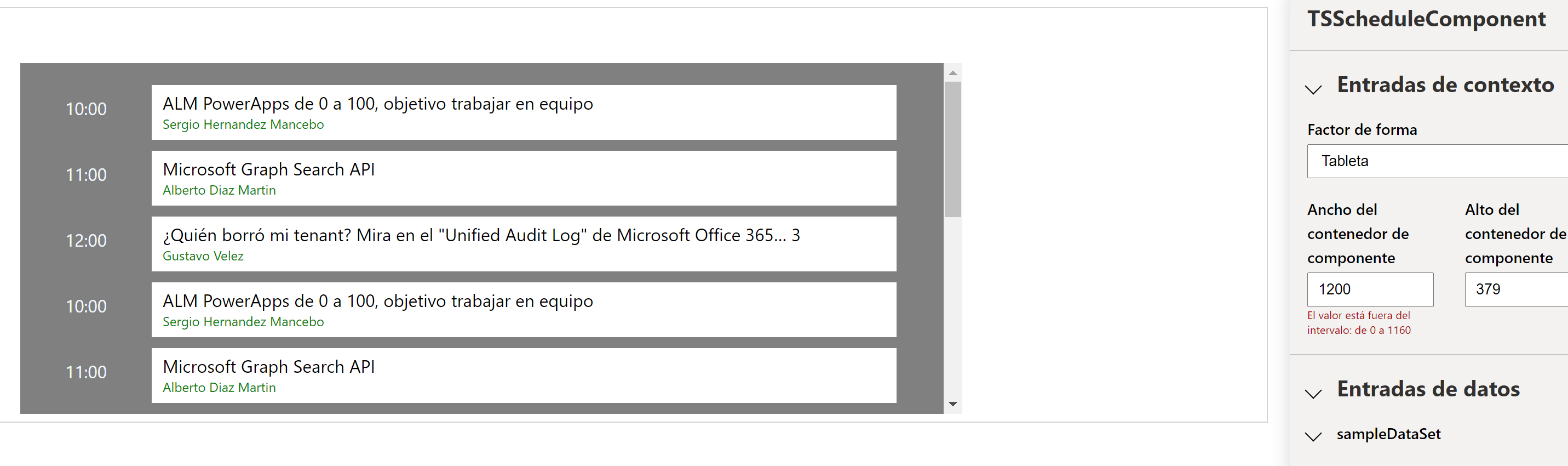


Para poder probar el componente bien, debemos seleccionar en “Entrada de datos”, y un ejemplo de fichero. Utilizaremos el fichero “testComponent.csv” que encontrareis en la solución descargada desde Github.

Una vez subido el csv, debemos revisar que las properties, están cruzadas de forma correcta con las columnas del fichero csv, y darle a cargar.



Si todo va como debe, ahora podremos visualizar el componente con los datos ya mapeados.



### **Desplegar el componente como una solución en Power Apps**

Necesitamos empaquetar el componente en una solución para poder importarla en nuestro entorno de trabajo .

Para esto vamos a crearnos una carpeta fuera del componente con el nombre **schedulesolution,** y nos vamos a posicionar en la consola sobre ella, para ejecutar el comando:

pac solution init --publisher-name developer --publisher-prefix dev

Una vez empaquetado la solución, vamos a añadir nuestro componente con:

pac solution add-reference --path <path del componente>

Esto habrá generado una serie de ficheros dentro de la carpeta con los xml de la solución.

Para poder generar el fichero Zip que encontraremos en la carpeta **schedulesolution\bin\debug** debemos ejecutar los comandos dentro de la carpeta schedulesolution:

msbuild /t:restore

msbuild

Una vez con el Zip podemos subir la solución a nuestro entorno como hemos hecho con la solución base.

### Subir la solución a nuestro entorno de Test

Para cubrir este último punto debemos hacer como antes, y en el apartado de soluciones seleccionamos en Importar.

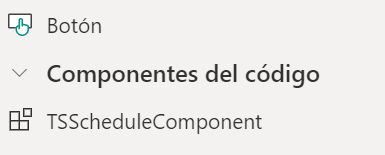
Una vez importada la solución podemos ver dentro de la misma el componente desarrollado.

### Integrar la solución en la App Event Management

Accedemos a la propia aplicación y en el menú Insertar->Personalizado, seleccionamos en “Importar Componente”.

En el menú de componentes, seleccionamos la pestaña código, y después el componente TSScheduleComponent

Desde ese momento en la biblioteca de controles lo tenemos disponible en el apartado de “Componentes de código”, tal cual se ve en la siguiente imagen.



Lo agregamos y configuramos las siguientes propiedades del componente, para que el data-set quede correctamente configurado:

|  |  |
| --- | --- |
| Propiedad | Valor |
| Items | Session |
| Schedule | Filter(Session;ThisRecord.'Schedule Session'.Day = Dropdown1.SelectedText.Value).Time |
| Session | Filter(Session;ThisRecord.'Schedule Session'.Day = Dropdown1.SelectedText.Value).Name |
| Speaker | Filter(Session;ThisRecord.'Schedule Session'.Day = Dropdown1.SelectedText.Value).Speaker |