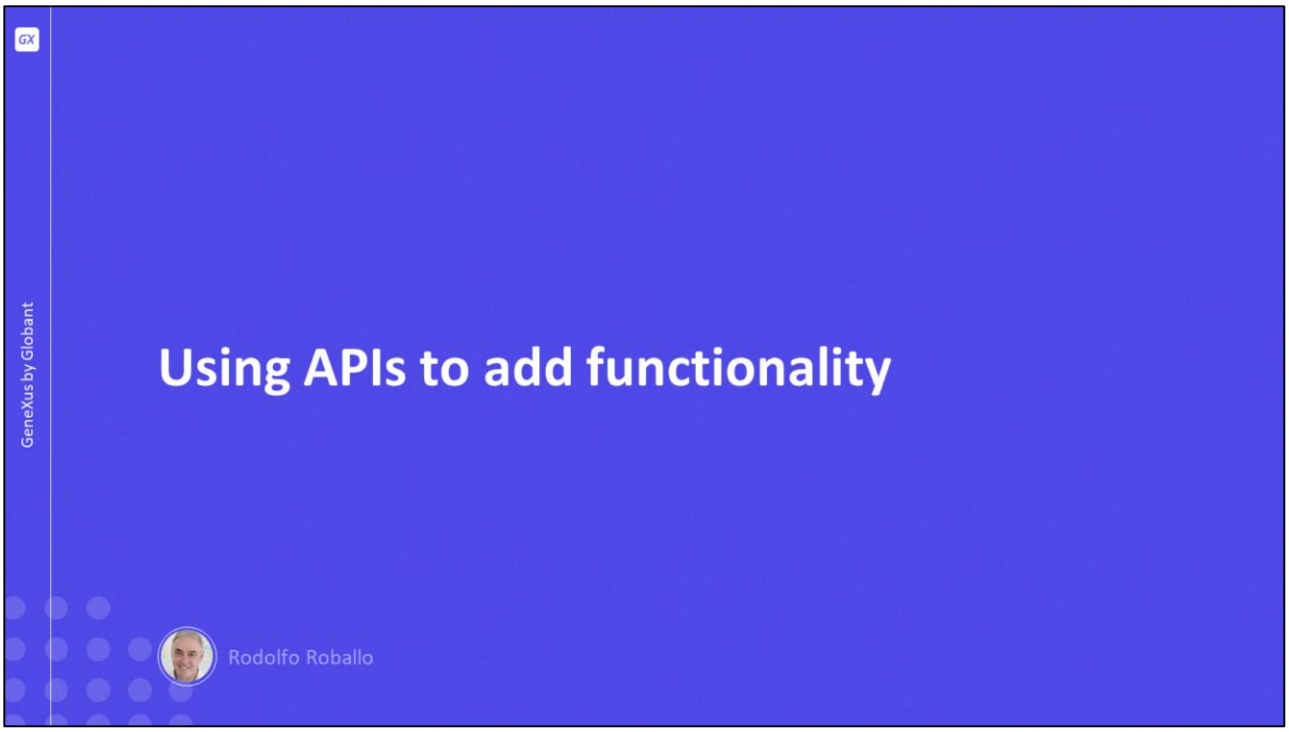
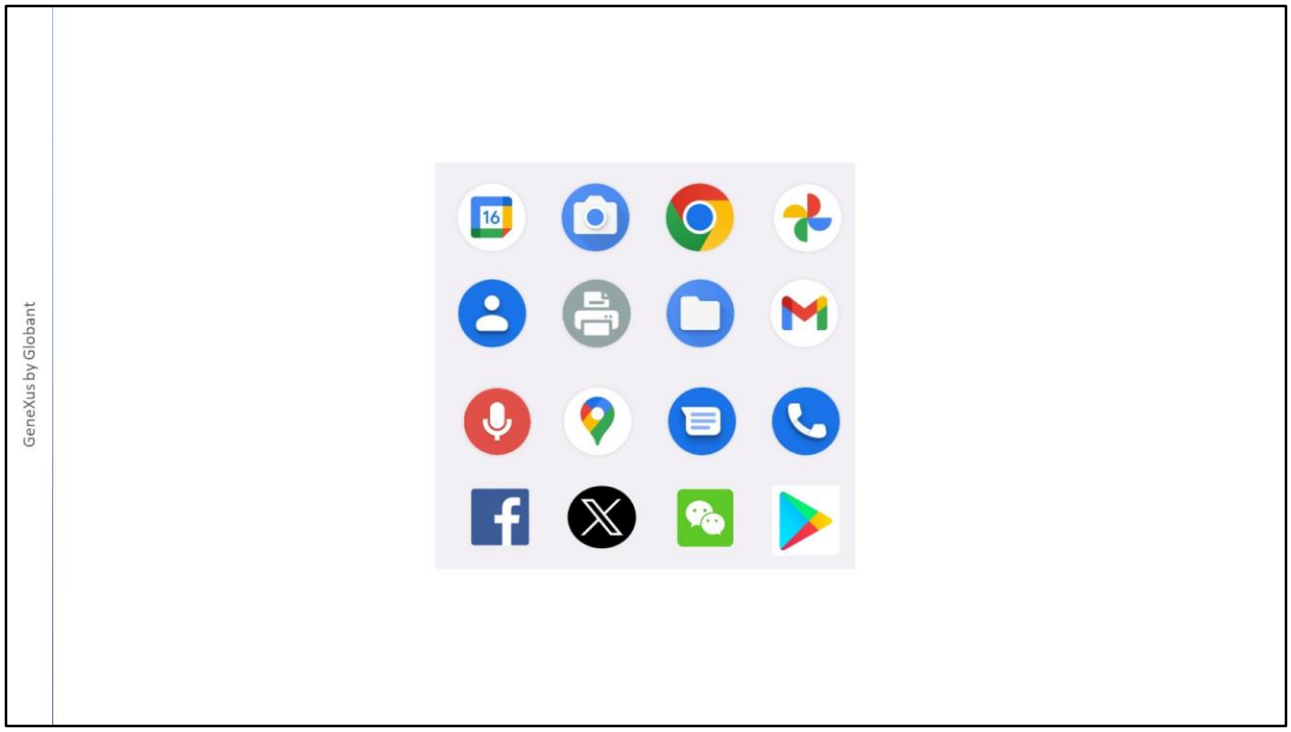



En este video veremos cómo agregar a nuestra aplicación, funcionalidades que nos permiten la integración con recursos del dispositivo, tanto físicos como lógicos, así como también facilitarnos la comunicación con aplicaciones externas.

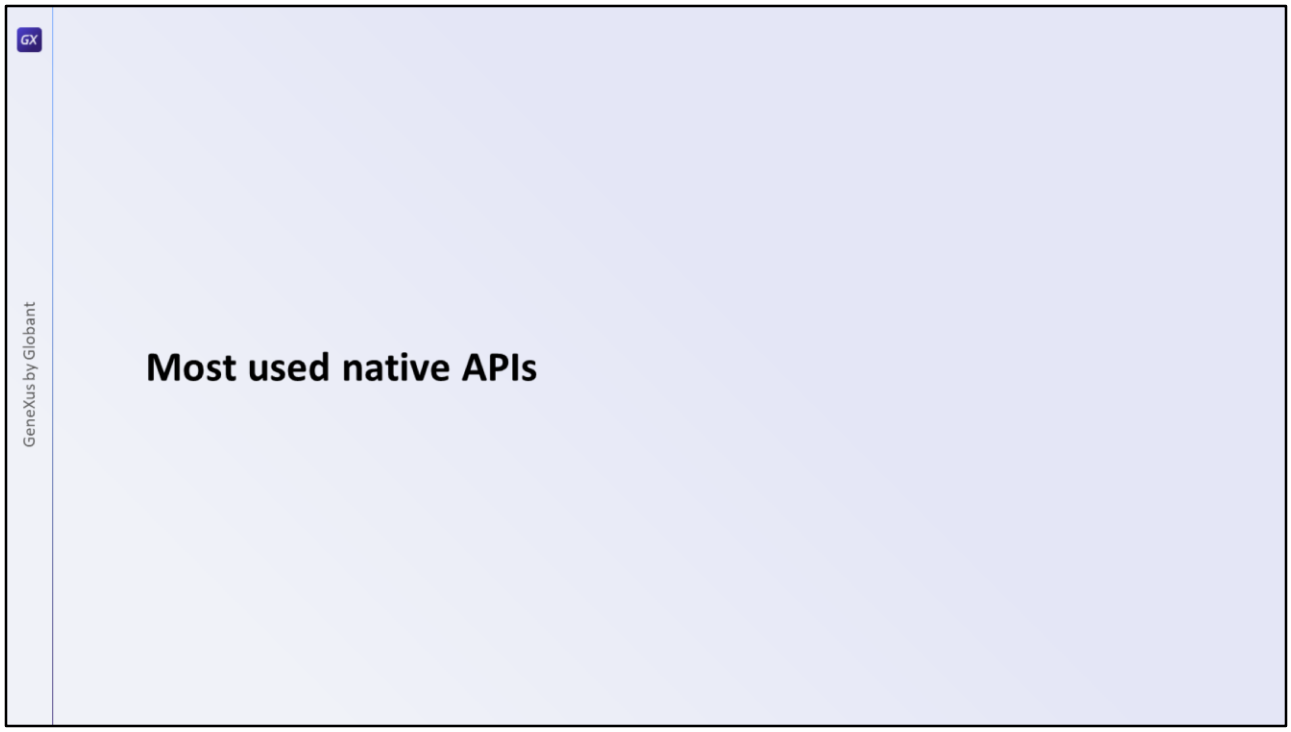



Mediante objetos externos predefinidos en GeneXus, podemos acceder programáticamente a muchas de las funciones que dispone un dispositivo físico, como usar la cámara de fotos, acceder a la grabadora de audio, hacer una llamada, o imprimir cierto contenido.

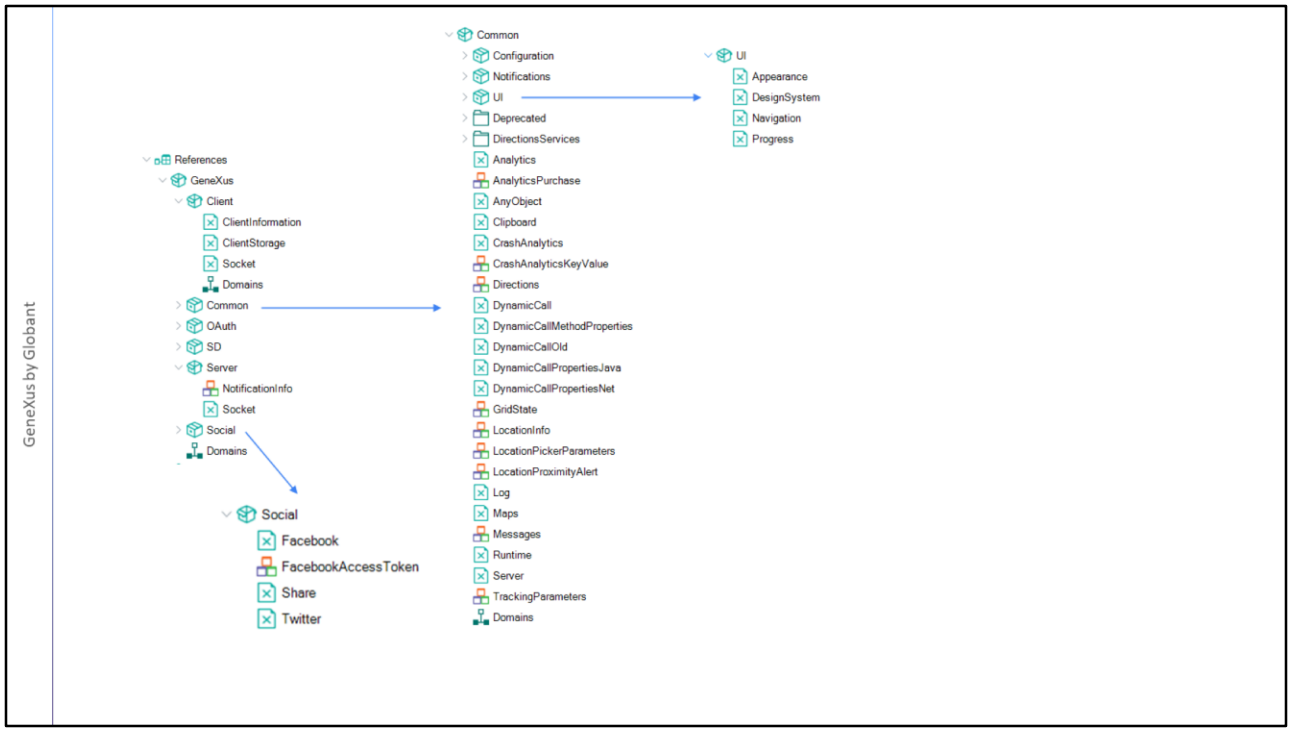
También podemos acceder a aplicaciones que ya vienen instaladas en el dispositivo, como el calendario, los contactos, acceso a los archivos, enviar un SMS, enviar un mail, abrir el browser o acceder al store.

Y también podemos interactuar con redes sociales como Facebook o Twitter.

A continuación veremos cómo acceder a algunas de estas funcionalidades.




Las APIs disponibles en GeneXus pueden ser para uso exclusivo en dispositivos móviles nativos, otras especiales para aplicaciones Web, y otras que valen para una y otra plataforma.



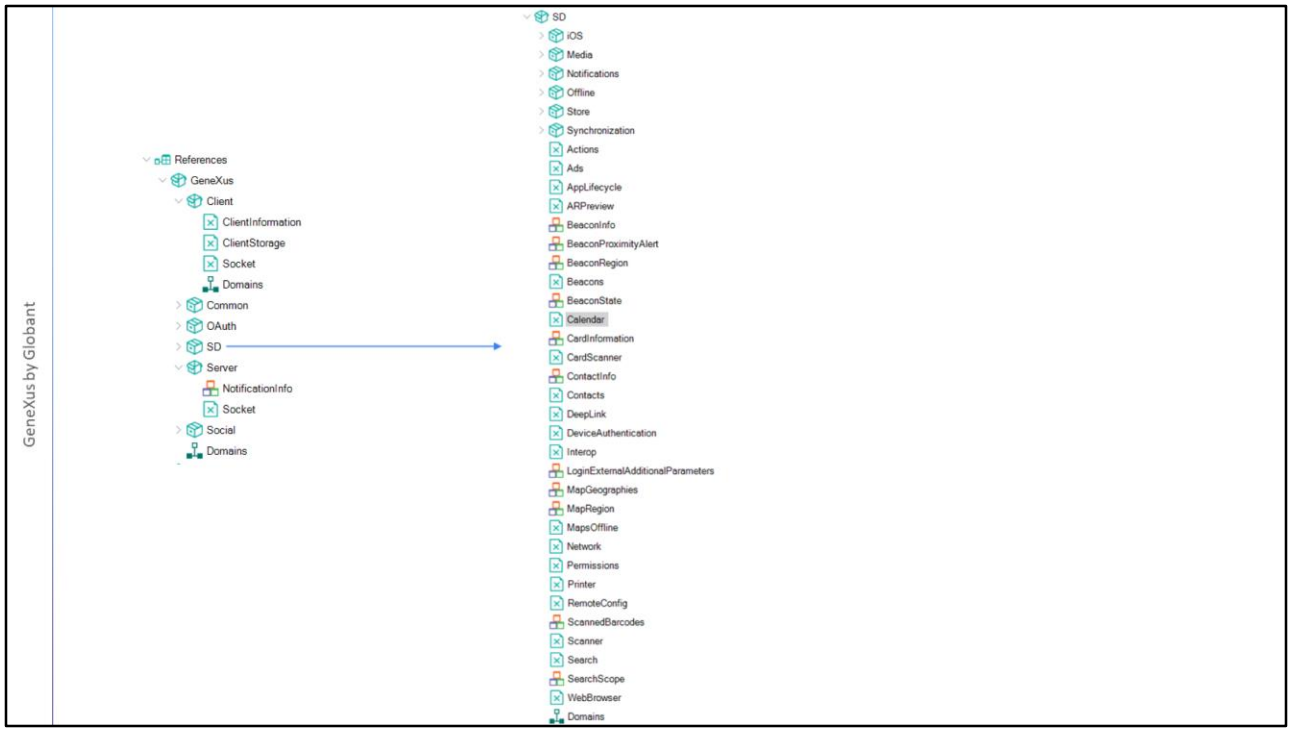

Estas APIs las encontramos en el KB Explorer bajo el nodo References, dentro del módulo GeneXus, donde además se encuentran todos los SDTs, dominios y Procedimientos que estas APIs requieren. Este módulo, junto con sus submódulos se crean cuando se crea la KB, y todos sus objetos ya están compilados, razón por la cual son read-only. Esto hace que no deban regenerarse cada vez que se hace build de la aplicación, lo que mejora mucho los tiempos de compilación.

Por otro lado, cualquiera puede desarrollar un módulo con objetos externos, dominios, sdts, etc. para implementar tanto funcionalidades nativas como para web y distribuirlo. Y vamos a poder importar esos módulos y utilizarlos de la misma forma en que utilizamos las apis predefinidas. De esta manera se facilita el trabajo cooperativo en la comunidad de desarrolladores.

Si echamos un vistazo rápido, vemos que tenemos apis especiales para SD, otras especiales para Web, y otras que valen para una y otra plataforma.

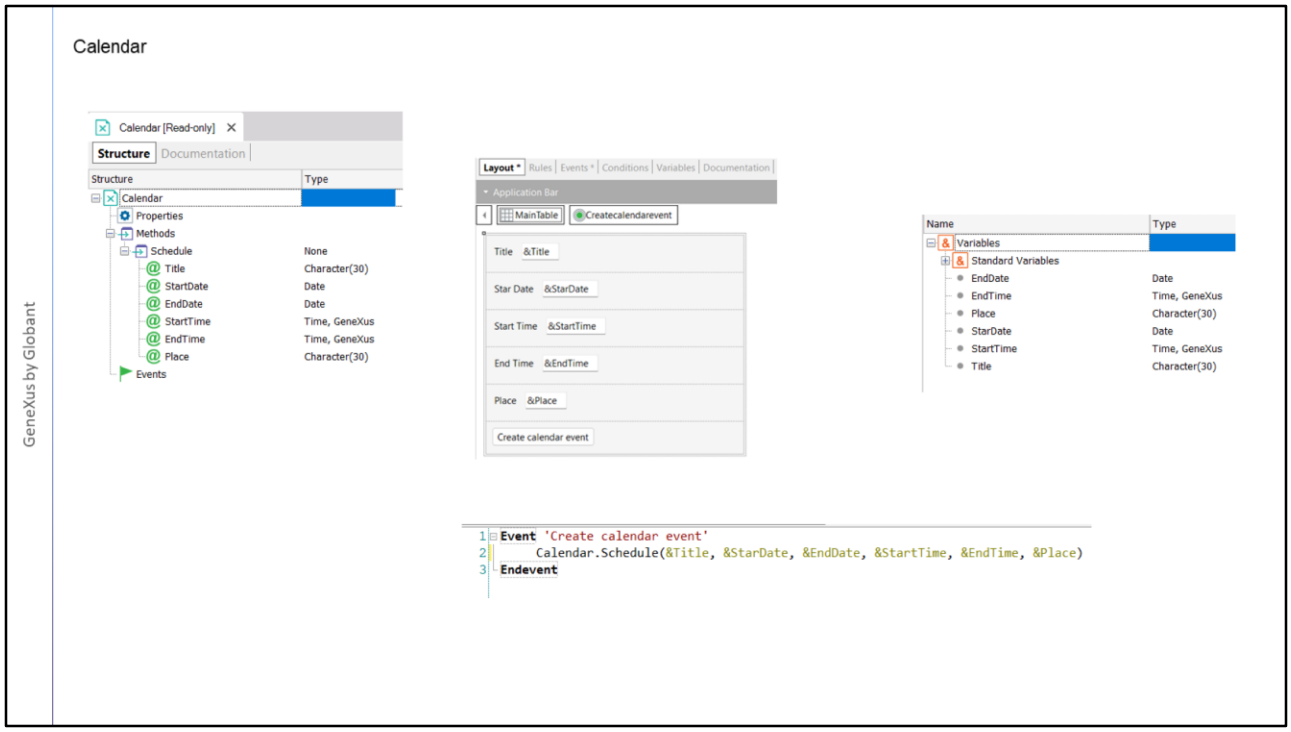
Por ejemplo, dentro del submódulo Common vemos la API Maps, la API Log o la API Analytics que podemos usar tanto en aplicaciones web como nativas; dentro del submódulo UI, user interfase tenemos el objeto externo Navigation, que nos permitirá mostrar regiones de la pantalla u ocultarlas o el objeto externo DesignSystem que nos permite que nuestra aplicación pueda configurar por código aspectos de diseño.

Dentro del submódulo Social vemos apis para interoperar con Facebook o Twitter, así como para compartir información con alguno de los programas instalados en el dispositivo.




Si abrimos el grupo SD, vemos todas las APIs que podemos utilizar en un dispositivo móvil nativo.

Veamos cómo agregar una cita al calendario.



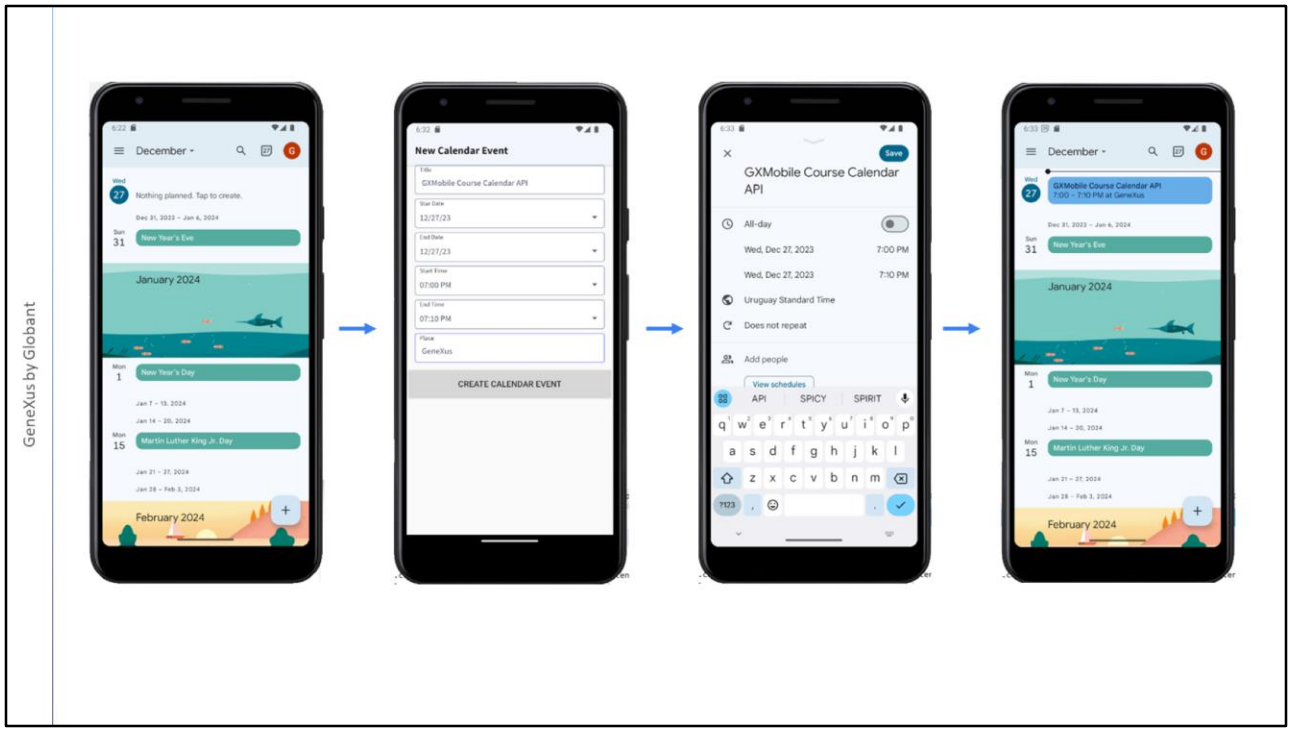

Si abrimos la API Calendar, vemos que no tiene propiedades y tiene un solo método, el método Schedule.

Si vemos sus parámetros, reconocemos la información necesaria para agregar una cita en el calendario, como el título, fecha y hora de inicio, fecha y hora de fin, y el lugar.

Una cosa importante es que tenemos que pasar todos los parámetros y en el orden en el que aparecen en el método, si hay un paramento cuyo dato no tenemos, dejamos vacío el lugar para ese parámetro, pero lo escribimos en la propia invocación.

En un panel de nombre NewCalendarEvent, tenemos en pantalla las variables necesarias para los datos que precisa el método de la API Calendar y un botón con el que llamaremos al método Schedule, pasándole los parámetros requeridos.

Vamos a setear este panel como main, y como ya tenemos el emulador abierto, ejecutamos el panel.



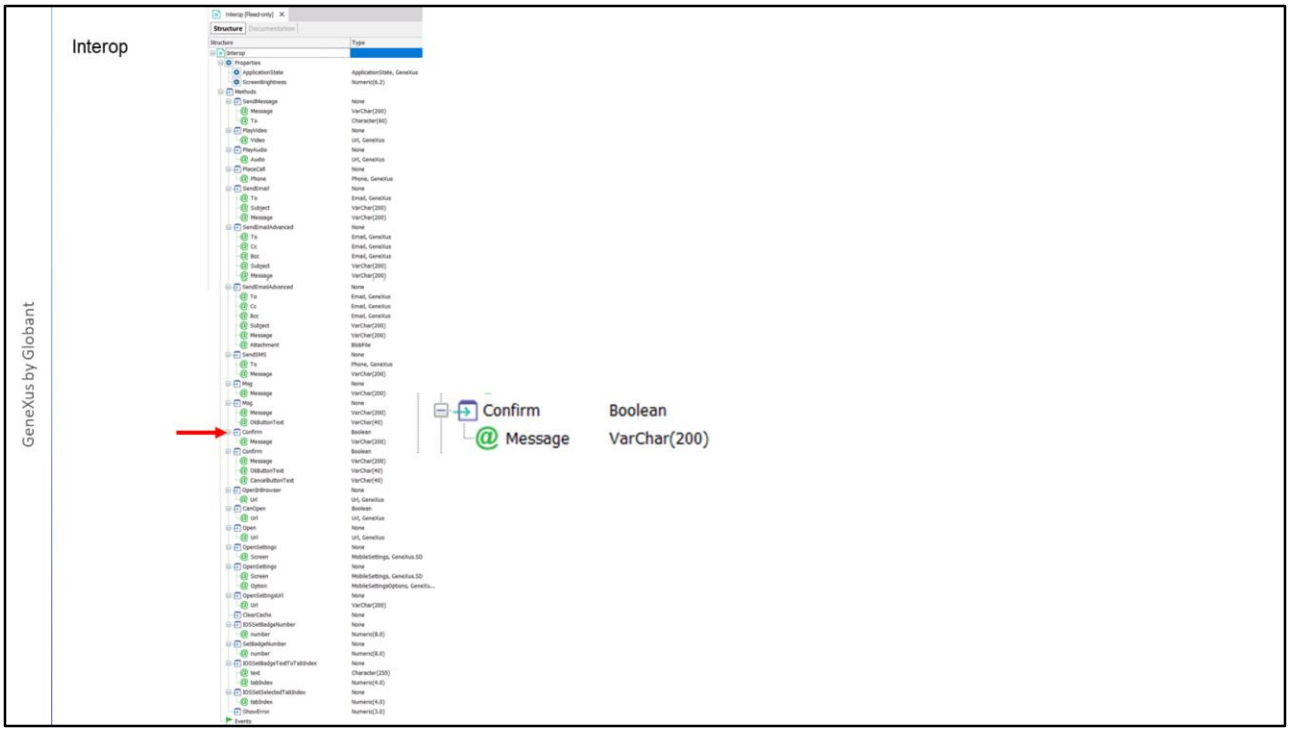

Vemos que se ejecuta nuestro panel. Antes de ingresar datos, vamos a la aplicación Calendario, nos logueamos y vemos que no hay agregado ningún evento, para el día de hoy.

Ingresamos un evento llamado GXMobile Course API Calendar para hoy, para dentro de una hora y que dure 10 minutos, en el lugar le ponemos GeneXus y presionamos el botón CREATE CALENDAR EVENT.

Vemos que en el dispositivo se abrió una ventana con los datos que ingresamos, está todo correcto así que presionamos Save. Y ahora volvemos al calendario y vemos que apareció el evento que creamos por código desde nuestra aplicación utilizando el Objeto Externo Calendar.

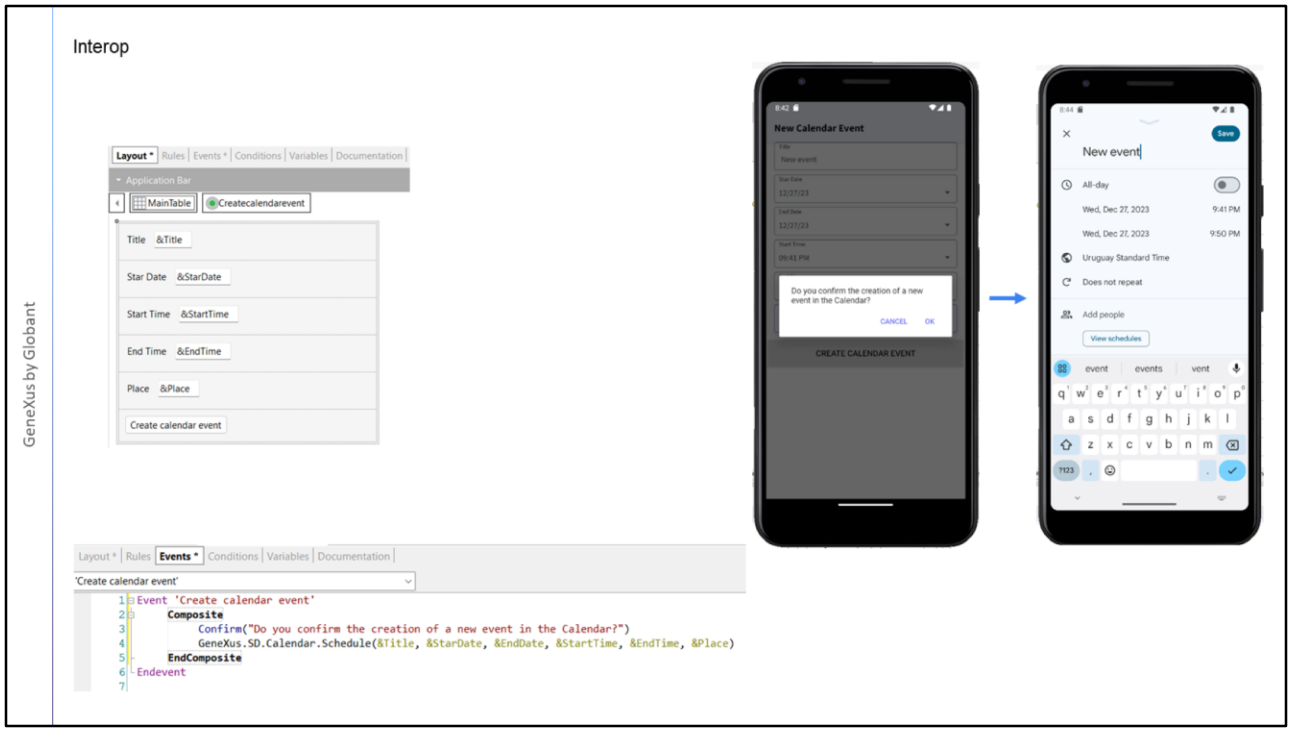

GeneXus dispone de algunos objetos externos que proveen acciones comunes, como ir directo a la pantalla principal, retornar a la pantalla anterior, salvar un contenido y otras que proveen las funcionalidades más comunes en un dispositivo móvil como enviar mensajes, confirmaciones, enviar correos o mensajes SMS.

Veamos un par de estas APIs multifuncionales, la API Interop y la API Actions.




Una API que es muy utilizada es la de nombre Interop y que ofrece la interoperabilidad con app de mensajería, permite enviar mensaje por mail y SMS, así como desplegar mensajes al usuario para informarlo de algo o para pedirle confirmación sobre una acción, ejecutar videos o audios y muchas otras funciones como se pueden ver en los métodos del objeto.

Vamos a utilizar el método Confirm para agregar una instancia de confirmación antes de agregar la cita de calendario que vimos en el ejemplo anterior.



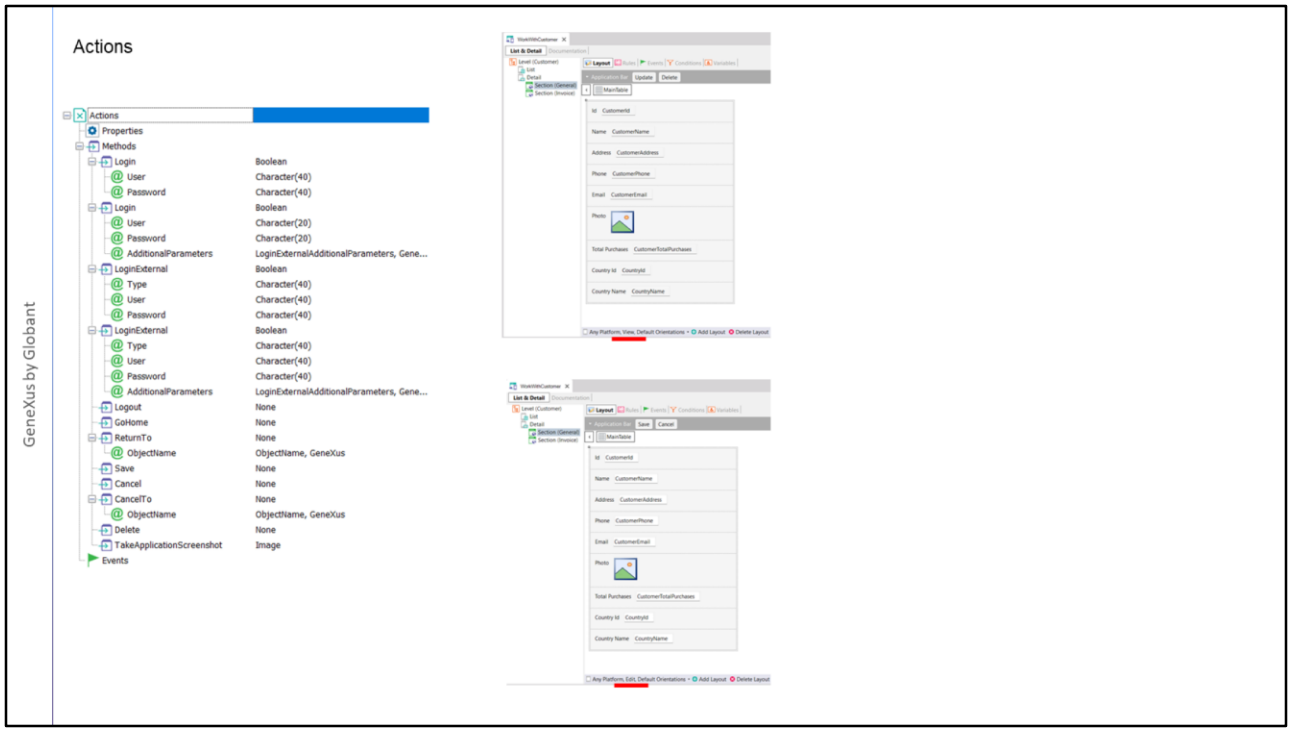

Para eso invocaremos al método Interop.Confirm pasándole como parámetro el mensaje de confirmación que deseamos que aparezca. Dado lo frecuente que va a ser la utilización del método Confirm, no necesitamos escribir Interop.Confirm, podemos escribir directamente Confirm y GeneXus lo entiende sin inconvenientes.

Vamos nuevamente al panel NewCalendarEvent y en el evento del botón podemos arrastrar interop punto y elegir el método Confirm o como decíamos, escribir directamente Confirm y luego entre paréntesis va el texto que queremos que se despliegue al usuario. Observemos que si bien el método Confirm retorna un valor booleano, no es necesario evaluar el valor retornado, porque GeneXus tiene la inteligencia de ejecutar o no la línea siguiente del código, dependiendo de si aceptamos o cancelamos la acción.
Recordemos que el bloque Composite es opcional, pero lo agregamos porque nos provee manejo automático de errores.

Ejecutamos…

Vemos que al presionar el botón, se abre una ventana emergente con el texto de confirmación que escribimos y tenemos la posibilidad de aceptar o cancelar.

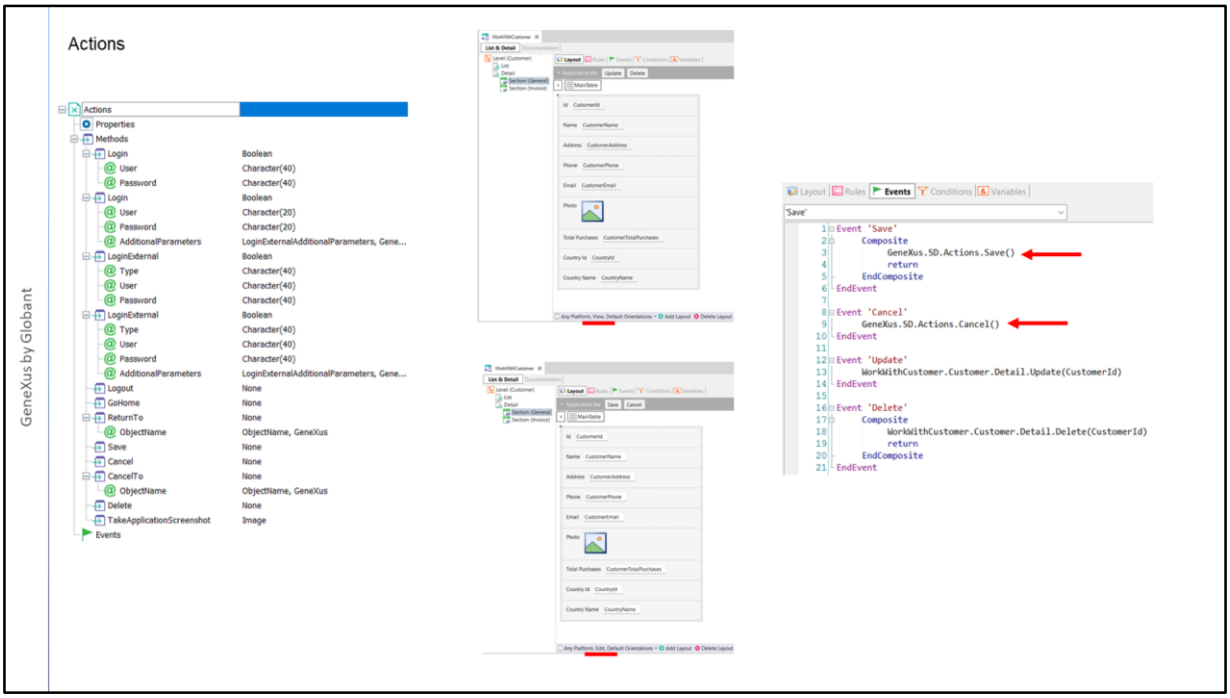
Si presionamos CANCEL se vuelve a la pantalla de ingreso de datos del panel y si presionamos OK, vemos que se abre la pantalla de creación del nuevo evento.




Hay una API que ya había aparecido anteriormente pero que no habíamos reparado en su funcionamiento que era la API Actions, que permite entre otras cosas abstraer la funcionalidad de Login, Save, Cancel, métodos, estos últimos, que ya habíamos visto en los eventos que implementaba automáticamente el pattern Work with a nivel del Detail.

Aquí vemos el objeto WorkWith que se crea al aplicar el patrón WorkWith a la transacción Customer.

Por ejemplo, veamos el nodo Detail, Section (General). Tenemos los botones Update y Delete para el layout del modo View, y para el layout del modo Edit, los botones Save y Cancel.




Si observamos su programación, vemos que el evento 'Save', que corresponde al caso en el que estamos editando la información de un cliente y estamos queriendo grabar esa información en la base de datos, utiliza el método Save de la API Actions.
El método Save de esta API lo que hace es encapsular la invocación al business component correspondiente, que es el que efectivamente realiza la inserción en la base de datos, recordemos que este Business Component es un servicio REST, una api que exponía nuestra aplicación en el servidor.

Podríamos preguntarnos porqué en la invocación no aparece solamente Action.Save(), sino que se usa todo el camino: GeneXus. Sd. Action. Save()
Recordemos que tanto GeneXus como SD son módulos y los objetos que estén dentro de ellos son visibles dentro del módulo pero pueden haber otros módulos dentro de la misma kb con objetos de igual nombre, pero que son objetos independientes. La manera de desambiguar esta situación, es utilizar todo el camino nombrando el módulo contenedor principal y luego los submódulos hasta llegar al objeto que queremos utilizar.

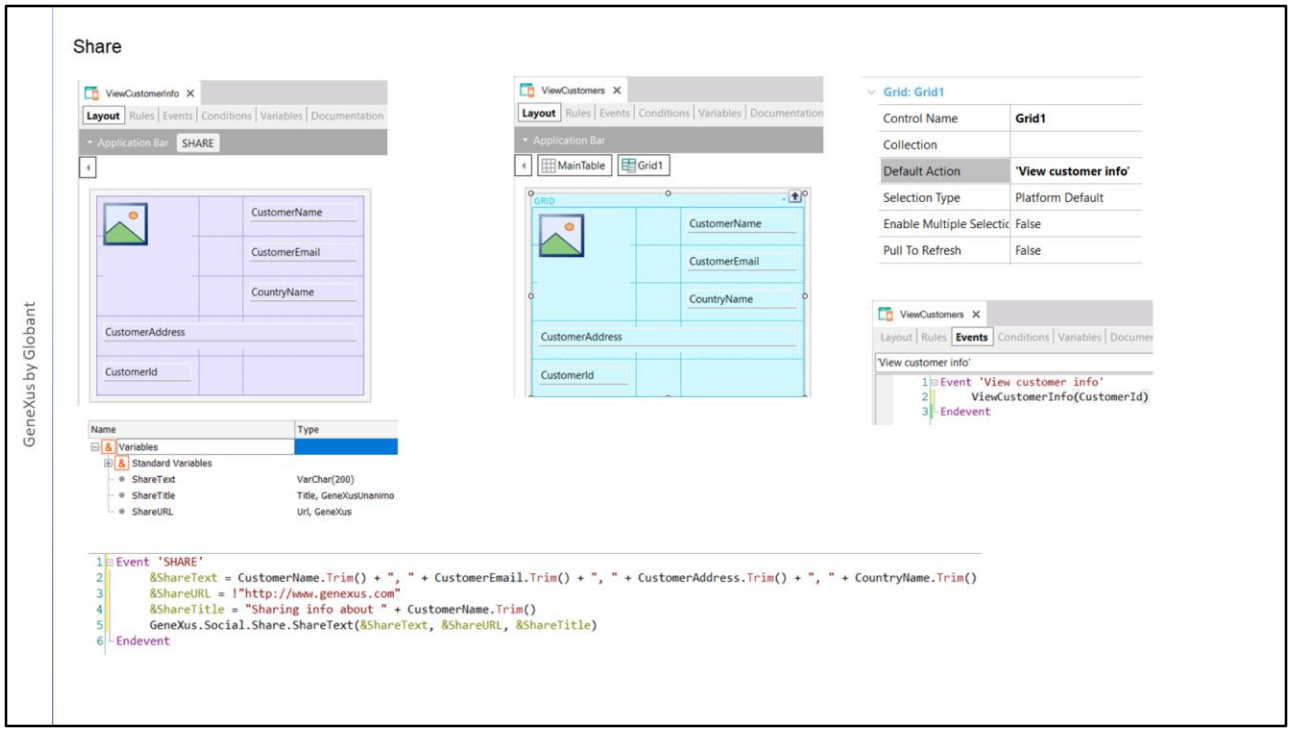
Esto depende de si tenemos otros objetos con igual nombre, por lo que dar todo el camino puede ser necesario o no. Sin embargo, es más seguro incluir siempre todo el camino, sobre todo en caso de que se incluyan APIs desarrolladas específicamente para el proyecto y si en el mismo trabajan muchos desarrolladores.

Volviendo al ejemplo del WorkWith, observamos que el evento Cancel utiliza el método de igual nombre de la API Action. Todo esto está programado de forma predeterminada en el patrón WorkWith, por lo que si aplicamos este patrón a otra transacción, observaremos una programación análoga de estos eventos utilizando esta API.


Dentro del módulo Social habíamos visto APIs para interoperar con Facebook, Twitter y WeChat, así como para compartir información con alguno de los programas instalados en el dispositivo
Veamos ejemplos de usos de estas APIs.


Vemos que cada API tiene propiedades y métodos de acuerdo a su funcionalidad. Por ejemplo la API Share tiene métodos que nos permite compartir un texto, una imagen o un archivo.

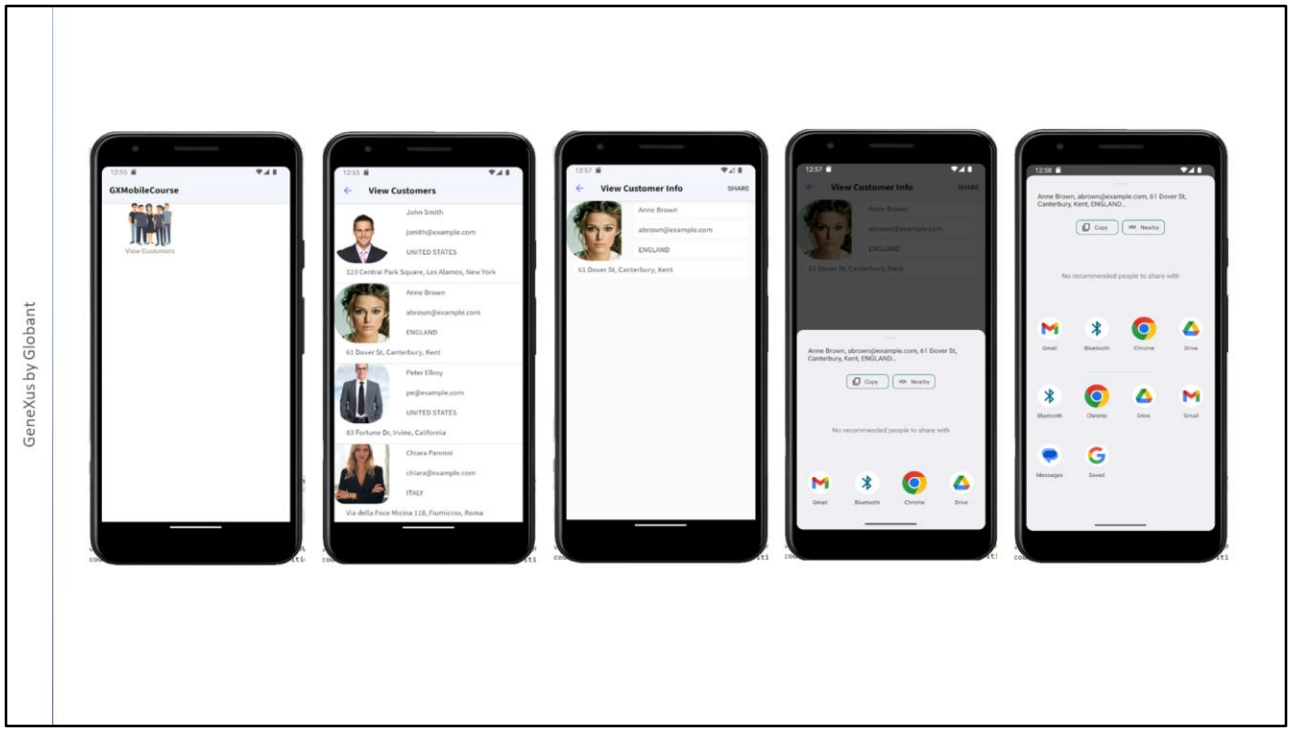
En el caso de las APIs de las redes sociales Facebook y Twitter, los métodos nos permiten realizar las acciones que se requieren para interactuar en estas redes, como el método PostToWall de Facebook o el método Follow de Twitter.




Veamos un ejemplo de uso de la API Share para compartir los datos de un cliente.
Aquí ya tenemos creado un panel ViewCustomerInfo que tiene los atributos de Customer y que recibe en la regla Parm al CustomerId por parámetro.
Ya tenemos agregado un botón SHARE a la Application Bar.
También tenemos definidas tres variables, una para cada parámetro, o sea una para el texto con los datos del cliente, otra para la url y otra con el título de lo que vamos a compartir.
En el evento del botón asignamos valores a estas variables y escribimos la invocación a la API Share, pasándole los parámetros requeridos.

En el panel ViewCustomers agregamos en la propiedad Default Action del grid, una acción con el nombre “View customer info” y en el evento correspondiente invocamos al panel ViewCustomerInfo pasándole por parámetro el identificador del cliente seleccionado.

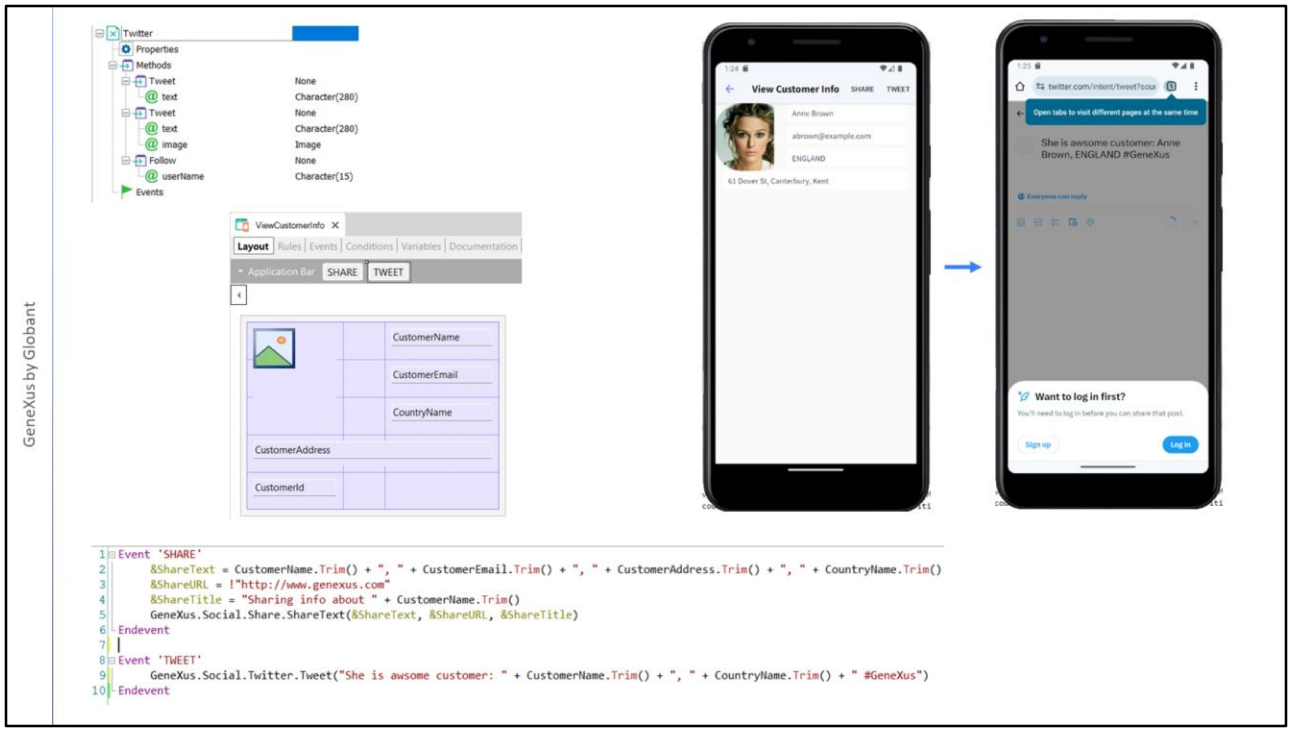
Ejecutamos….




Damos tap sobre el icono View Customers y luego damos tap sobre la cliente Anne Brown.

Vemos que se abre el panel con los datos de Anne y arriba a la derecha aparece el botón SHARE.

Si lo presionamos… Vemos que nos aparece los datos de la cliente para compartir y las aplicaciones que nos ofrece el dispositivo en las que podemos usar esta información o compartirla.




También podríamos querer ofrecer la acción de “twittear” un mensaje haciendo referencia al cliente seleccionado.

Para implementar esto utilizamos la API Twitter y el método Tweet. Si observamos la API nos ofrece dos métodos Tweet: en uno solamente enviamos el mensaje a ser “tuiteado” y en el otro el texto y la imagen. Y por otro lado tenemos el método Follow.

Volvemos al panel ViewCustomerInfo, agregamos un botón TWEET a la Aplication Bar y en el evento del botón invocamos el método Tweet de la API Twitter, pasando por parámetro el texto que queremos twittear con el hashtag al final.

Ejecutamos… elegimos de nuevo a Anne Brown y vemos que se abre el brower en la página de Tweeter que nos pide loguearenos, pero igual se puede ver el mensaje que twiteamos. Si hubiéramos tenido la app de Twitter instalada, se nos ofrecería enviar mensaje directo, o escribir un tweet general.

Podríamos querer hacer lo mismo, pero con Facebook, para lo que utilizaríamos la API correspondiente.


En este video vimos la cantidad de APIs que disponemos para distintos requerimientos y mostramos algunos ejemplos de uso.

A medida que GeneXus avanza, estarán disponibles más objetos externos con nuevas funcionalidades, por lo que les sugerimos consultar la documentación del wiki en forma frecuente