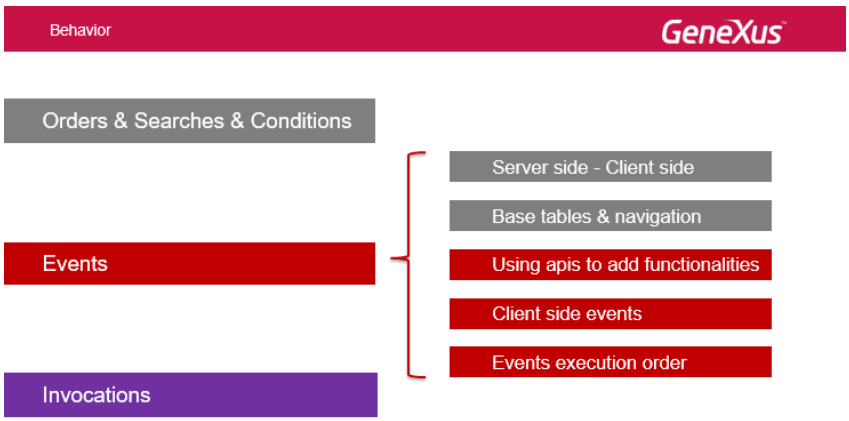
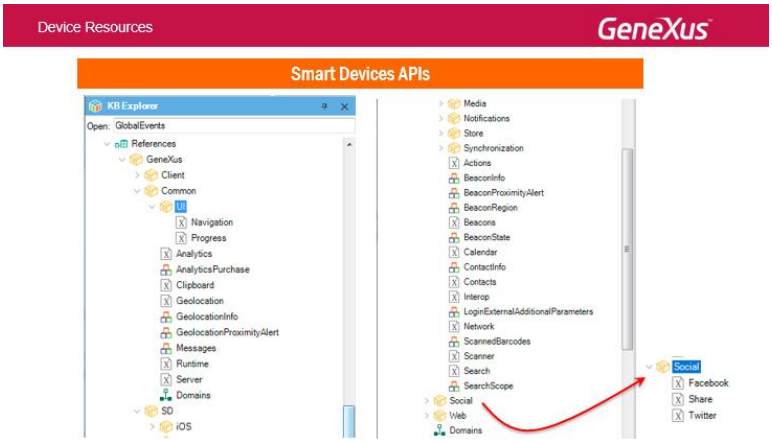
Events Using APIsTo Add Functionalities



Continuaremos con la forma de agregar funcionalidades (por ejemplo aquellas que permiten integrarse con el dispositivo) a través de apis.

Las aplicaciones para dispositivos móviles requerirán integrarse con recursos del dispositivo, tanto físicos como lógicos.

Bien, en definitiva, GeneXus nos brindará un conjunto de APIs, en forma de objetos externos, para poder agregar diferentes funcionalidades a la aplicación móvil y en particular, para permitir la integración con otras aplicaciones y funcionalidades del dispositivo.



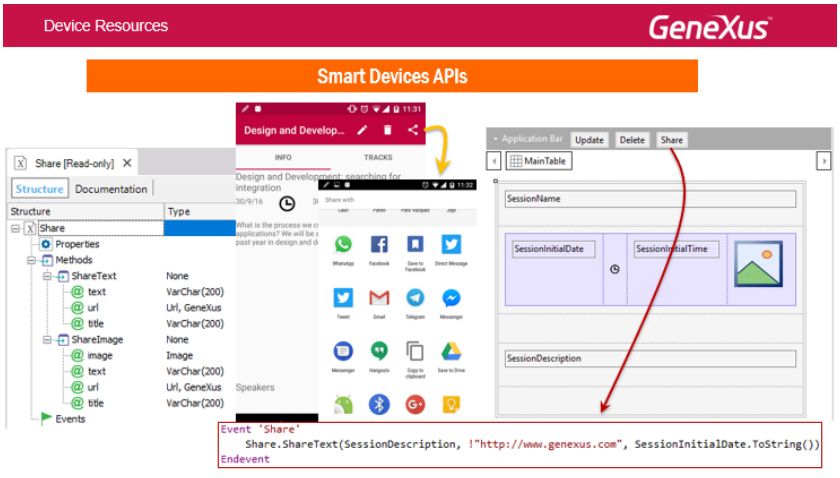
Es dentro del módulo GeneXus bajo el nodo References que se encuentran todas las apis, SDTs, y Procs que estas apis requieren.
Recordemos lo que ya habíamos dicho al principio del curso, cuando creamos de cero la KB: este módulo, junto con sus submódulos viene con la KB, y todos sus objetos ya están 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. Lo mismo va a suceder con los dominios predefinidos.

Por otro lado, cualquiera puede desarrollar un módulo con objetos externos, procedimientos, sdts, para implementar tanto funcionalidades SD (y también web) y distribuirlo. De esta manera vamos a poder importar esos módulos y utilizarlos de la misma forma en que utilizaremos las apis predefinidas. De esta manera se va a facilitar el trabajo cooperativo.

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.

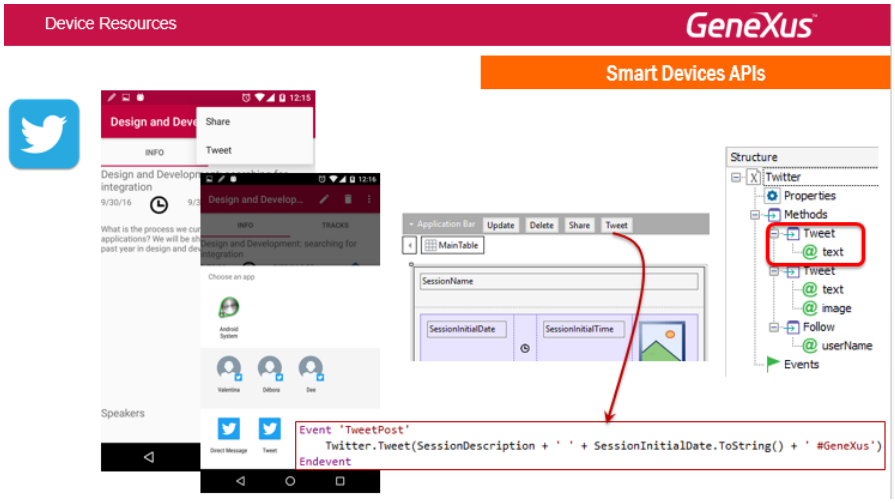
Por ejemplo, dentro del submódulo Common vemos la apis Geolocation; dentro del submódulo UI, user interfase tenemos el objeto externo Navigation, que nos permitirá mostrar regiones de la pantalla u ocultarlas. Dentro del módulo Social vemos apis para interoperar con Facebook y con Twitter, así como para compartir información con alguno de los programas instalados en el dispositivo…



…como por ejemplo podemos ver aquí, donde queremos que el usuario pueda compartir a través de alguna de

las apps que tiene instaladas, la descripción y la fecha de la conferencia que está visualizando.
Por ejemplo, puede decidir compartirla por Whatsapp, o Facebook, o Twitter, por mail, etc.

¿Cómo lo implementamos? Agregamos un botón “Share” en la application bar de la session(General) del Work With de Sessions lo estamos viendo aquí y programamos el evento asociado, llamando al método ShareText ofrecido por la api de nombre Share, estamos viendo aquí a la izquierda



Vamos a verlo en GeneXus… (Antes observemos que como solo tenemos 3 botones hay espacio para que aparezca en la application bar y por eso está apareciendo el icono, observemos, de share)
Vamos a verlo decíamos en GeneXus y ya en mi dispositivo…

Aquí estamos entonces en la transacción Session, en el Word whit for Smart devices, sessions General of Delete.
Vemos aquí el botón de Share, si vamos a la programación del evento asociado, vemos que está llamando entonces al objeto externo share, que lo podemos ubicar como decíamos bajo el nodo references, submodulo Social, que si expandimos aquí encontramos estas tres apis (objetos externos). Abrimos el Share y vemos que está ofreciendo primero que es read only entonces esos dos métodos. De los cuales nosotros vamos a usar el primero, Sharetext pasando estos tres parámetros (y eso es lo que vemos aquí).

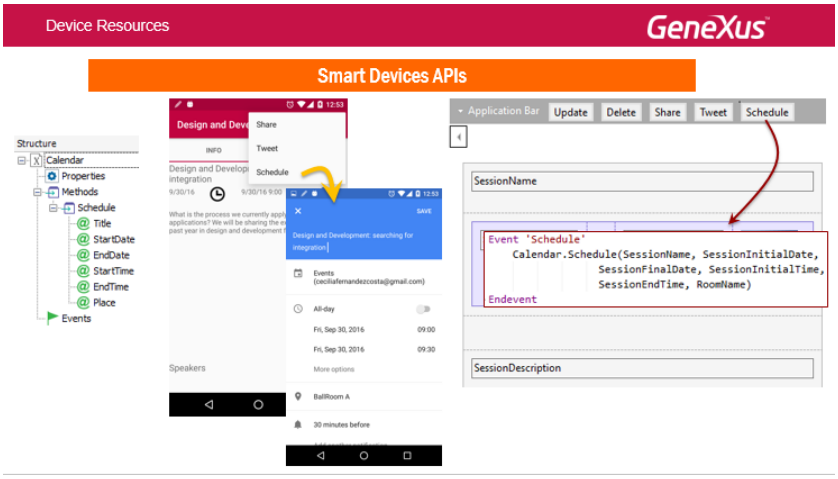
Vamos a verlo en GeneXus, yo tengo conectado el dispositivo, entonces vamos a elegir ver las Session, voy a elegir una, por ejemplo esta, vemos que tenemos una serie de acciones, la primera de las cuales es la acción de share, de compartir, entonces si ahora la elijo, vemos que me ofrece todos los programas que tengo instalados para poder compartir la información de esa conferencia. Por otro lado, voy a volver, no voy compartir, ya aprovechamos y mostramos que también tenemos dos acciones más, que se ofrecen en ese menú, una es para permitir twittear la información de la conferencia, y la otra para agendar la información de esa conferencia en el calendario de mi dispositivo.

Si elijo twittee vemos que se abre, me ofrece escribir un mensaje directo en twitter o un twittee, y vean que ya está abriéndose la aplicación de twitter con la información de la conferencia que va a ser la que nosotros le pasamos, está siendo la descripción y el hashtag GeneXus, además de la hora.

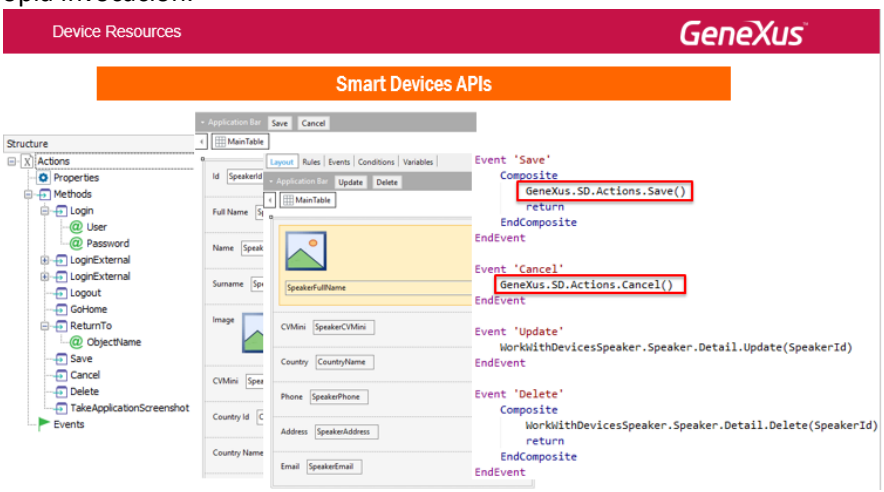
Y la otra alternativa decíamos, era utilizar la acción de agendar en mi calendario, la aplicación vean que me está abriendo mi app de mi Calendario en mi dispositivo, pasando para agendar el lugar la conferencia.

Bueno, en el caso de twitter decíamos, vemos que ahí agregamos el botón tweet, al que le asociamos el evento

Tweet post que vemos, y estamos invocando a la api objeto externo Twetter, método tweet, vemos que tenemos dos posibles métodos para llamar del mismo nombre twett que van a variar en la cantidad de parámetros que le pasamos, en este caso solo un parámetro que es el texto queremos que aparezca entonces cuando se abra la aplicación de Twitter para que el usuario pueda ya partir de ese texto o personalizarlo.



Por otro lado, decíamos que tenemos también la api Calendar, que tiene un solo método, el método Schedule, que es el que estamos usando, en el evento asociado al botón Schedule pasando la información, los paramento que nos pide el método.
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.



Una api que había aparecido anteriormente pero sobre la que no habíamos reparado era la api **Actions**, que permite entre otras cosas abstraer la funcionalidad de Login, Save, Cancel, métodos, estos últimos, que ya habíamos visto aparecer en los eventos que implementaba automáticamente el pattern Work with a nivel del Detail como estamos viendo en la pantalla.

Por ejemplo, vamos a ver el Detail de Speakers, section (General). Tenemos los botones Update y Delete para el modo View, y para el Edit, los Save y Cancel.

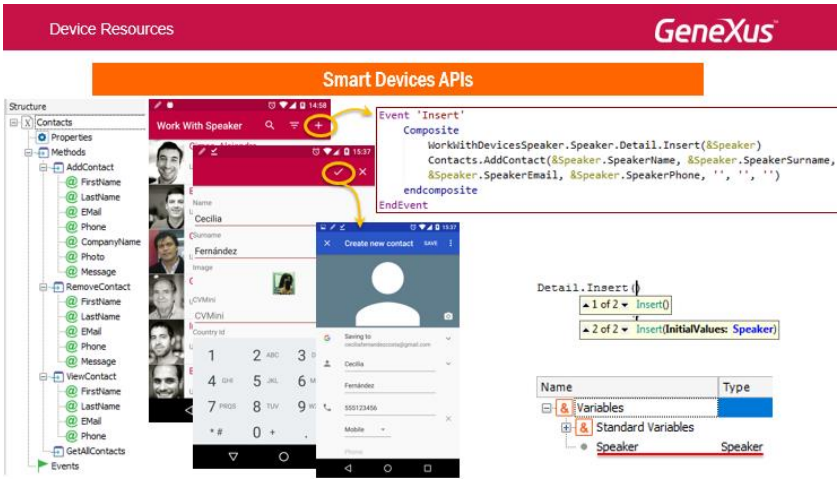
Si observamos su programación, vemos que el evento 'Save', que corresponde al caso en el que estamos editando la información de un orador y estamos queriendo grabar esa información en la base de datos, utiliza el

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 es eso es un servicio rest una api que exponía nuestra aplicación. Luego, con el return, se retorna al llamador.

¿Por qué en la invocación podemos preguntarnos no aparece sólo Action.Save?
Porque está apareciendo GeneXus. Sd. Action. Save?
Recordemos que estos son módulos y el asunto con los módulos es que los objetos que estén dentro de ellos son visibles dentro del módulo pero pueden haber otros módulos dentro de la misma kb con objetos que reciban el mismo nombre son otros objetos, entonces la manera desambiguar de decir que objeto externo action estoy llamando si hubiera otros es a través dela información del caminito de los módulos.
Esto dependiendo si tenemos otros objetos con otros nombres o no va a ser necesario o no, por eso nosotros en los pasos anteriores cuando lo tuvimos que programar nosotros directamente no pusimos todo el camino donde se encuentra el objeto externo calendar, por ejemplo, llamamos directamente a ese objeto.

Bien también observamos que el evento cancel utiliza el método de igual nombre de la api, esto se hacía todo de forma predeterminada si voy a GeneXus observemos que aquí estamos en la section general del Session, en la ppts hablábamos de speakers pero es lo mismo. Vemos que tenemos el Update y Delete para el modo view y para el modo Edit teníamos los botones de Save y Cancel, y los eventos se programaban en una única pantalla, podíamos variar, teníamos dos lyouts diferentes pero la ventana de eventos es la misma, por eso aquí están programados los 4 y vemos entonces lo que estaba apareciendo en las ppts, las invocaciones.

Por otro lado, vemos este bloque Composit, en composit, que lo vamos a ver en la clase siguiente, pero que ya podemos adelantar que se trata de un bloque obligatorio cada vez que debamos escribir más de una línea de un comando dentro de un evento en el cliente.



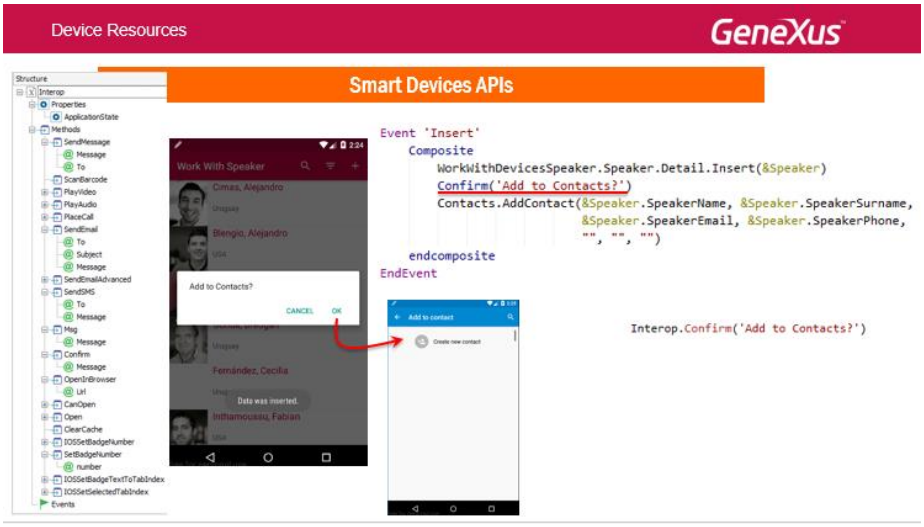
Supongamos ahora que desde el List de Speakers queremos que tras insertar un nuevo orador en la base de datos se lo agregue también como contacto en la app de contactos del dispositivo.
Para ello: después de invocar al Detail en modo Insert, cuando el usuario presiona aquí para dar por buena toda la información que ingreso se va a volver a quien lo llamo que fue el evento asociado a este botón y lo que queremos llamar (ahí abrir) es la aplicación de contactos para ingresar la información del speaker recién ingresado en esa libreta de contactos, entonces tendremos que hacer uso de la api Contacts que viene con GeneXus, en particular usaremos método AddContact para agregar un nuevo contacto , al que debemos pasarle esta serie de parámetros, sino tenemos alguno de esos datos pasamos la cadena vacía y lo invocamos de la manera que decíamos el nombre del objeto. Y el método.
¿Cómo hacemos para recuperar estos datos, los datos que debemos pasarle al método para enviarlas a la api?

Recordemos que estamos llamando primero al work with devices speakers detail en modo insert cuando estamos posicionados en este objeto, llamamos entonces a ese panel, el usuario va a ingresar todos los datos y se va a volver y tenemos que tener una manera de recuperar entonces los datos ingresados por el usuario para poder pasárselos a la appi contact.
Recordemos que el insert ofrecía dos opciones: la primera era no pasar ningún parámetro y la segunda era pasar un único parámetro del tipo de datos el business component asociado al nivel en el que se está queriendo realizar la inserción. Este Business Component nos permitía inicializar valores en la pantalla a la que estamos invocando, recordemos estamos desde aquí invocando a esta pantalla entonces enviando ese business component va a ser este business component de aquí en este caso si antes de enviarlo inicializamos alguno de esos valores al llamar a esta pantalla esos valores van a venir inicializados y el usuario va a tener que simplemente insertar los que falten o por supuesto modificarlos, y lo otro que va a pasar es que cuando desde esta pantalla se confirme la información y se vuelva a la que lo llamo entonces que ese business component va a venir cargado, va a volver cargado con toda la información cargado con toda la información cargada por el usuario. Y esa es la manera que tenemos de obtener entonces todos los datos que tenemos que pasarle al métodos addcontact.

Vamos a verlo ya implementado en GeneXus, vamos a la transacción speaker, método Insert en el nivel list vamos a ver el evento y aquí es que tenemos programado entonces, la invocación que ya venía predefinida al detail en el modo insert hemos creado una variable speakers del tipo de dato business component speakers para no verlo todo hecho vamos a ver como invocamos a una appi y uno de sus métodos y como GeneXus nos ofrece ayuda para ir ingresando los distintos parámetros, en este caso es contact si lo buscamos en el modo referens está aquí, entonces arrastramos por ejemplo si no recordamos exactamente el nombre y con punto vean que nos ofrece todos los métodos que tiene disponible ese objeto externo y en este caso vamos a elegir add contacts y vemos como nos va ofreciendo información de cada uno de los parámetros que tenemos que insertar en cada oportunidad, el método además nos va a devolver un valor buleano que podemos utilizarlo para algo o no, ponemos suprimir porque ya lo hicimos.

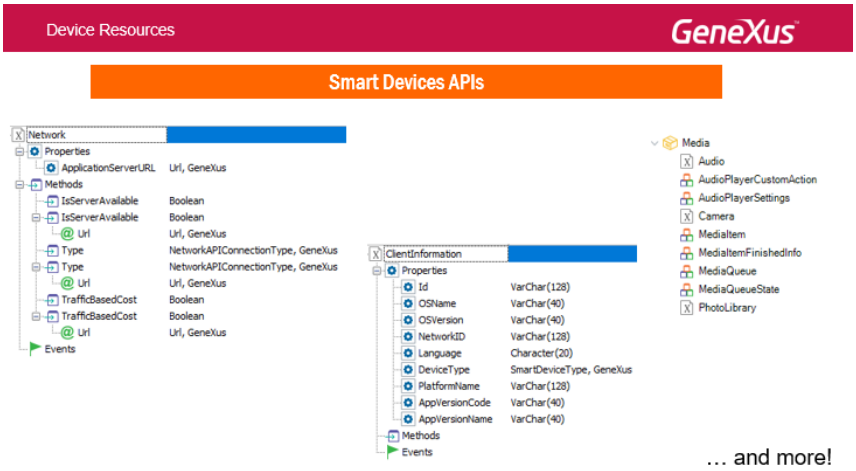
Y ahora vamos a probar en GeneXus que es lo que sucede entonces cuando ejecutamos el insert cuando insertamos un nuevo orador en la aplicación.
Vamos en la aplicación en el dispositivo, vamos a agregar un nuevo orador, vemos que aparece la pantalla vacía sin datos, voy a ponerme a mí como oradora, Cecilia Fernández, me pide una foto no voy a poner foto, podría elegir una de la galería de la imágenes o tomarme una foto, no voy a sacar una ahora, país que es necesario sino es una clave esporanea no me va a dejar grabar y vamos a agregar solamente el teléfono.
Vamos a ver qué pasa entonces al dar tap en el confirmar, vemos que está llamando abriendo la aplicación de contactos para que yo pueda crear un nuevo contacto y vemos que está saliendo la información que le pasamos con el teléfono y si grabamos se va a grabar esa información ese contactos y además si agrego está saliendo el nuevo orador (sin foto porque no se la puse) en el list.

Otra api que va a ser muy utilizada es la de nombre Interop que estamos viendo a la izquierda que ofrece la interoperabilidad con app de mensajería y permiten mandar mensaje por mail, sms así como desplegar mensajes al usuario para informarlo de algo o para pedirle confirmación, ejecutar videos o audios, etc.
En el ejemplo queremos pedirle al usuario que confirme antes de abrir la app de contactos, que confirme que desea agregar ese orador como contacto.



Observemos que por lo frecuente que va a ser la utilización del método Confirm, no necesitamos escribir Interop.Confirm. escribimos directamente confirm y GeneXus lo entiende de esa manera.

Vamos a probarlo en GeneXus bueno y agregamos entonces después de la invocación al detail y antes de la invocación al método addd contact podemos arrastrar interop punto y elegir el método confirm o como decíamos escribir directamente el Confirm y acá adentro lo que va es el texto que queremos que se despliegue al usuario.
Entonces vamos a probar, si yo estuviera usando live editing directamente podría probar esto en el dispositivo sin tener que hacer un Ram como me quedo deshabilitado lo voy a hacer.

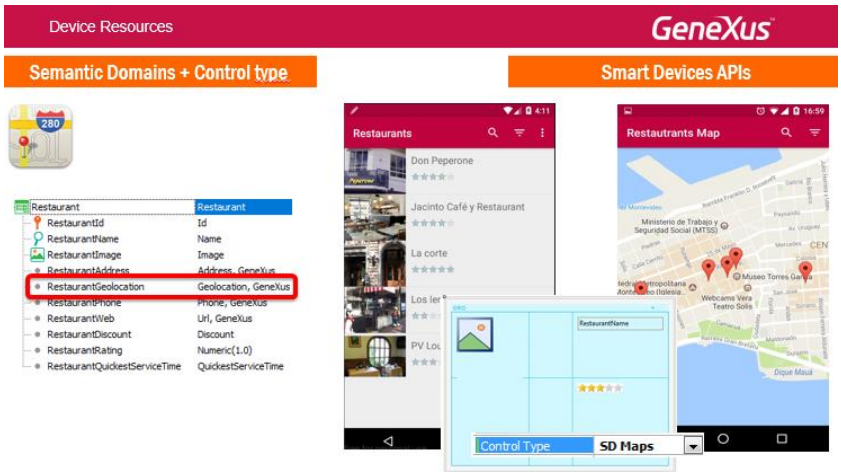


Tenemos unas cuantas apis más.

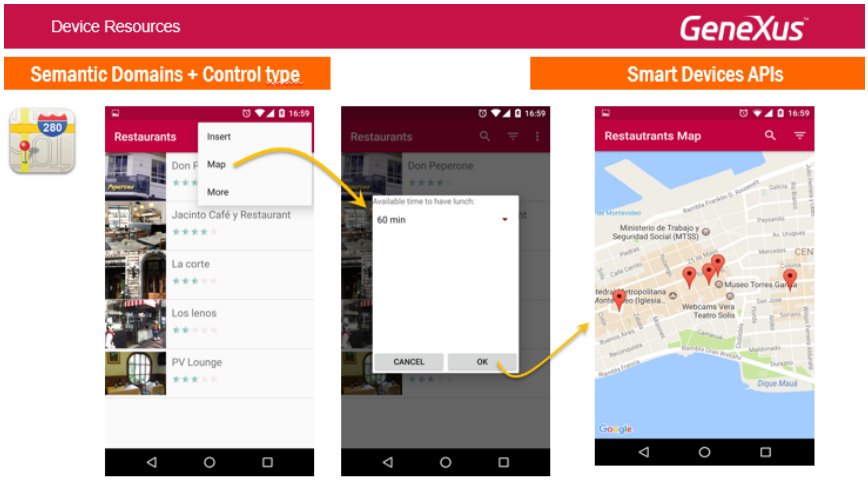
Por ejemplo, una que nos permite saber si hay conexión con el servidor en un momento dado, y el tipo de conexión otra, ClientInformation, para obtener la información del dispositivo que está ejecutando la aplicación (su id, sistema operativo, versión, lenguaje, tipo de dispositivo).

Algunas para lograr la reproducción de audio, listas de reproducción, interactuar con la cámara del dispositivo, haciendo que tome una foto o inicie la grabación de un video, una para interactuar con la biblioteca de fotos, y muchas más.

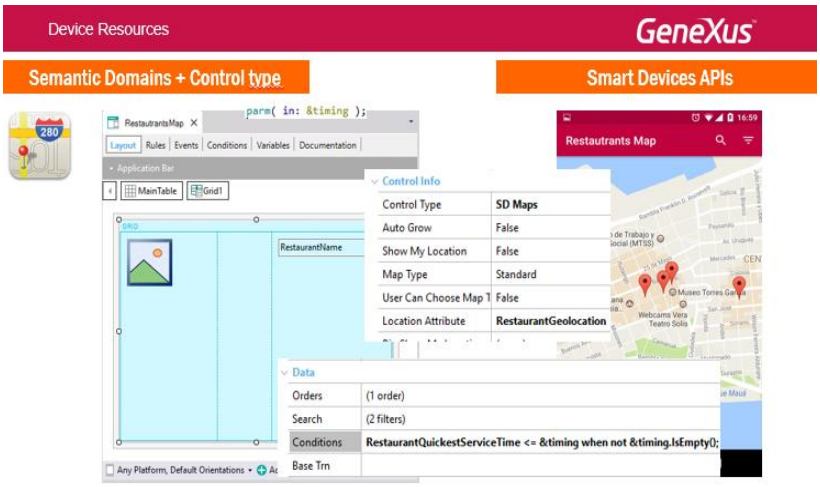
Por ejemplo, una muy utilizada hoy, para trabajar con la ubicación geográfica…



Ya usábamos el dominio semántico Geolocation cuando queríamos presentar los restaurantes no como un list estándar, sino como puntos en un mapa…. Para ello aviamos visto nos alcanzaba con cambiar el tipo de control del grid por SDMaps



En nuestra aplicación, tenemos implementado el List de Restaurants, que corresponde al nodo del Work With donde el Grid tiene el Control Type default… y por otro lado, vamos a ofrecemos al usuario la posibilidad de, eligiendo el tiempo del que dispone para almorzar, mostrarle los restaurantes que se comprometen a brindarle el servicio dentro de ese tiempo. Este es un panel que estamos viendo a la derecha cuyo layout es muy similar al del List del Work With, con la diferencia de la manera de mostrar la información del grid, que va a ser como puntos en un mapa, debido a que tenemos el atributo RestaurantGeolocation.



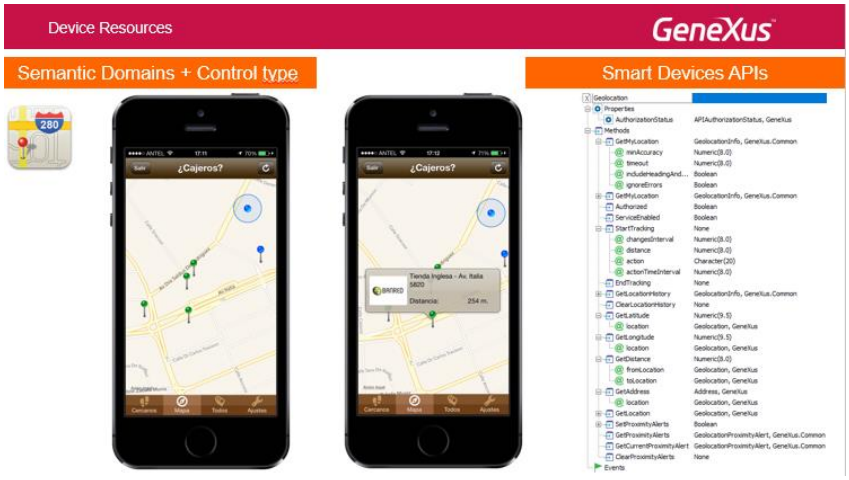
Esto lo conseguimos recibiendo por parámetro en ese panel que decíamos el tiempo de servicio Timed y filtrando el grid que se mostrará como SD Maps, de acuerdo al atributo RestaurantQuickestServiceTime, que registra el tiempo máximo en el que el restaurante se compromete a servir al cliente.

Si lo vemos en GeneXus, estamos aquí en la transacción Restaurant en el Word with, tenemos el nodo en la list en la acción bar el botón map, vamos a el evento asociado y vemos que hemos programado la invocación (esto lo vamos a ver más adelante por ahora no le prestamos atención, las call option) es más lo que podemos hacer es llamar a este panel restaurant filder al que le pasamos 2 parámetros timed y okey.
Si observamos ese panel vamos a abrirlo, abrimos el layout, vemos que son 2 variables dos parámetros de salida, van a volver cargados al llamador, si vamos al layout vemos que el usuario entonces va a ingresar el tiempo máximo que dispone para almorzar y cuando presione okey, vamos a los eventos, va a cargarse esa variable okey en true indicando que efectivamente el usuario ingreso un valor y se retorna al llamador y en caso que se presione cancel va a quedar en false y se cancela también retornando al llamador.
Bien entonces si vamos de nuevo al restaurant, vemos que si el usuario no cancelo la operación vamos a invocar a un panel for sd, vamos a abrirlo, restaurant maps, que como vemos es un grid que tiene las imagen del restaurant (restaurant image) y el nombre del restaurant va a mostrar esa información, si vemos sus propiedades vemos que tenemos en control type SD maps location attribute Restaurant Geolocation, vemos en Restaurant su estructura para recordarla y acá tenemos el tiempo este en el que el restaurant se compromete a bridar el servicio, entonces vamos de vuelta para aquí, y lo que vamos a ver es que como había recibido por parámetro ese tiempo máximo, si vamos a la sección grupo data del grid vemos que tenemos programadas las Conditions esta condición de aquí que el tiempo tiene que ser menor igual al tiempo que ingreso el usuario, entonces se van a cargar solamente los restaurantes que cumplan esta condición, y se van a mostrar como puntos en un mapa.

Si lo vamos a probar en GeneXus… vamos a hacer ran.

Vamos a los restaurant, allí elegimos la opción maps, vemos que se nos abre el panel intermedio que nos pide el tiempo máximo que disponemos para almorzar, supongamos que 90min, damos ok aquí abajo, nos está abriendo ese otro panel como puntos en un mapa los restoranes con los que cumplen con la condición deseada.

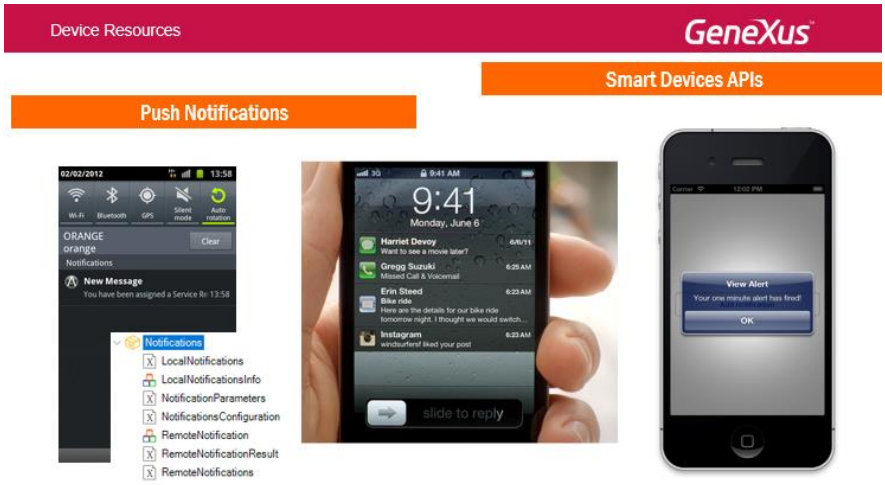
Ahora bien, podríamos solamente querer mostrar al usuario los restaurantes que se encuentran a menos de cierta distancia de la localización actual del dispositivo…



Allí es donde interviene la api Geolocation, que ofrece métodos para por ejemplo, obtener la geolocation actual -para lo cual necesita el gps del dispositivo-, realizar un tracking del camino que va realizando el dispositivo a lo largo del tiempo, finalizar el tracking, obtener la latitud a partir de una geolocalización, o la longitud, así como obtener la distancia entre dos puntos, u obtener la dirección a partir de una geolocalización, o viceversa, configurar alertas de proximidad, u obtenerlas, etc.

Por tanto, para mostrar solamente los restaurantes a menos de, supongamos, 400 metros de la ubicación actual de nuestro dispositivo, tendremos que usar el método GetDistance, y el método GetMyLocation, ambos lo vamos a usar en el Grid, a la hora de filtrar la información que mostramos.

No es ni más ni menos que lo que hace una aplicación para obtener los cajeros automáticos cercanos al punto donde nos encontramos como estamos viendo en las pantallas.

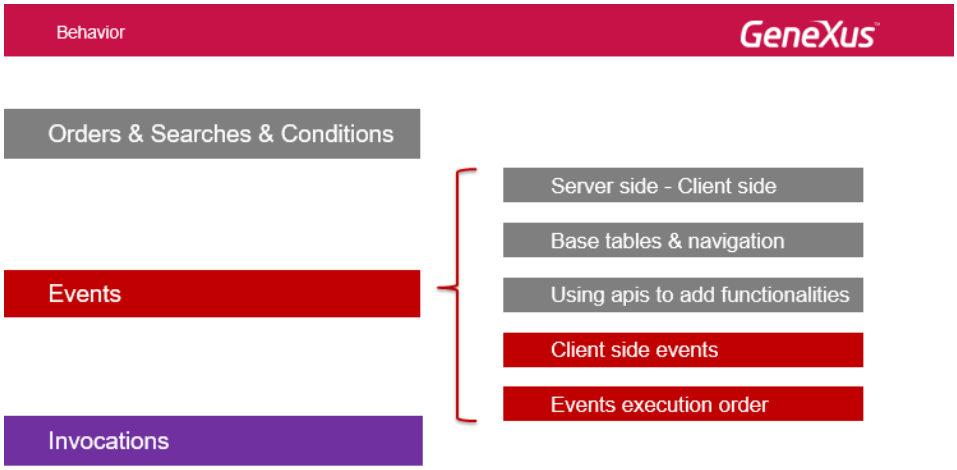


Otra funcionalidad muy utilizada corresponde a las notifications, que pueden enviarse desde el servidor web a los distintos dispositivos que tienen instalada nuestra aplicación móvil, incluso cuando la app no está corriendo.

De esta forma, el usuario recibirá la notificación que le envía el servidor y podrá realizar la acción que corresponda.
Para ello entonces vemos que vamos a tener un módulo de notificación con todas esas apis que estamos viendo.

Hemos presentado sólo algunas de las múltiples apis para Smart Devices que tenemos disponibles y además tengamos en cuenta que podemos trabajar colaborativamente creando nuestros propios módulos y y objetos externos y compartiéndolo con la comunidad.

Para obtener información de las demás apis que acabamos de mostrar y de las demás que existen le recomendamos acceder a nuestro wiki.



Ahora vamos a pasar al siguiente tema …