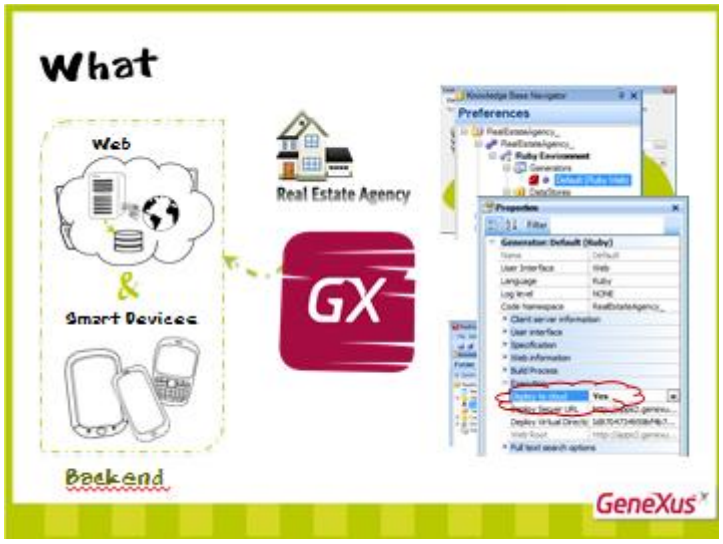
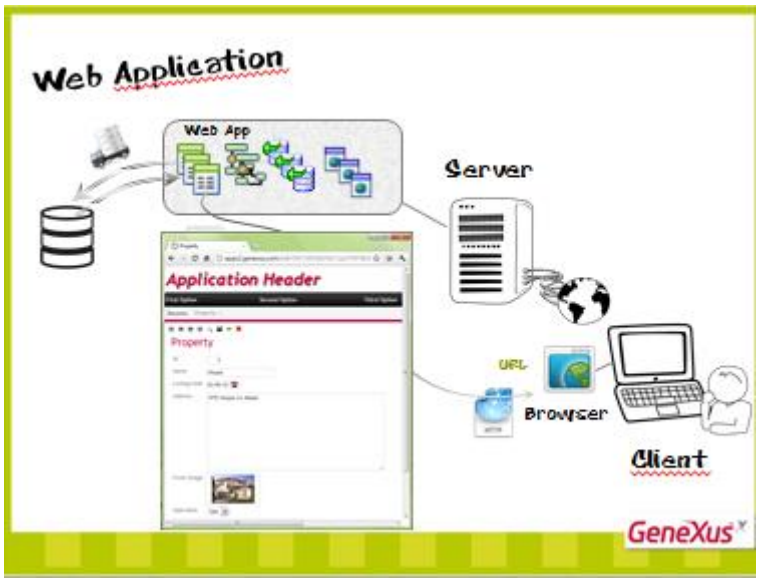
Arquitectura de aplicaciones para Smart Devices



Nuestro problema a resolver será construir un backend para una inmobiliaria: con una parte web y otra para Smart Devices, para ser utilizada por los agentes inmobiliarios en su trabajo móvil.

Para ello creamos una KB, la transacción Property para registrar las propiedades inmobiliarias en venta o alquiler, y definimos que el backend web se generará en ruby, en la nube.

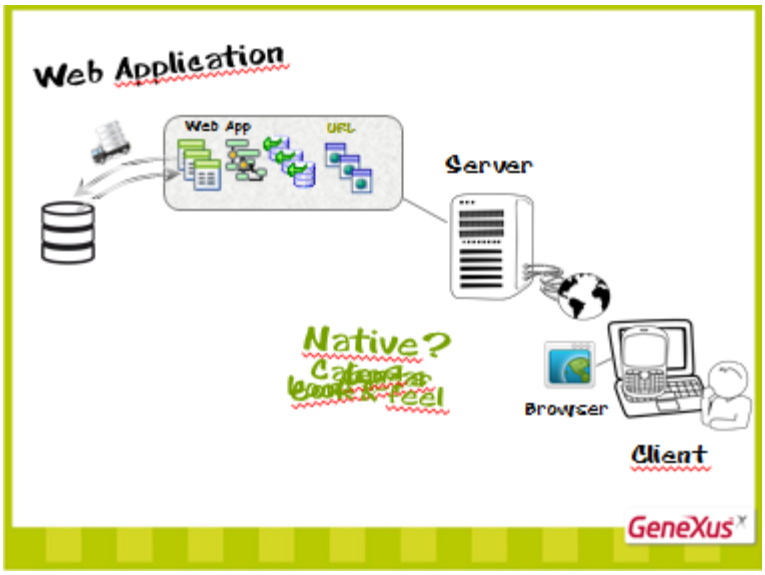
Para implementar la parte para Smart Devices, aplicamos el pattern Work With for Smart Devices a la transacción Property, y creamos un objeto Dashboard como punto de entrada. Decidimos en principio generar sólo en Android, la plataforma por defecto.



Para pensar la arquitectura subyacente en las soluciones para Smart Devices con GeneXus, partamos de lo conocido: las aplicaciones Web.

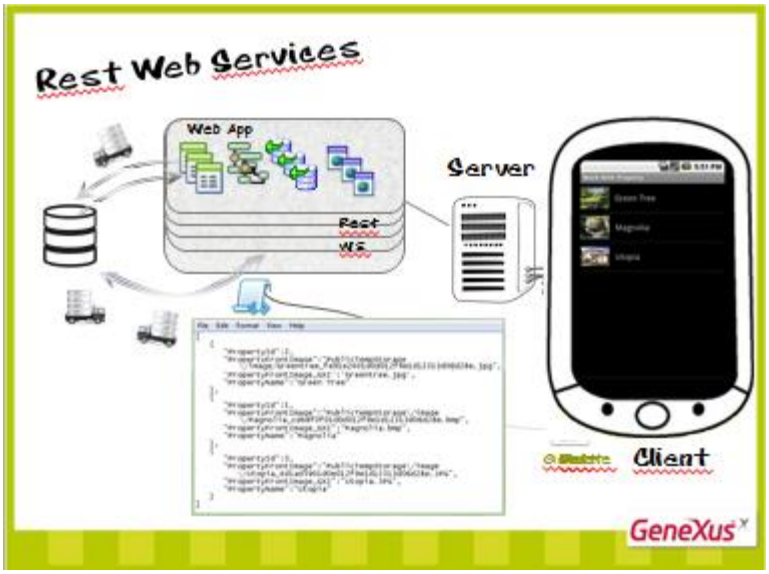
Tenemos por un lado un Servidor y por otro un Cliente. En el servidor tenemos la aplicación web y en el cliente un Browser.

Ejecutamos la aplicación Web a partir de una URL, por ejemplo la de la página home, web panel, o –a modo de ejemplo- la de la transacción para ingresar propiedades inmobiliarias. Probablemente el objeto requerido deba consultar la base de datos. Luego devuelve la información al cliente, para que el Browser arme el layout (HTML) que presentará al usuario como respuesta a su pedido.



Una posibilidad sería ejecutar la aplicación en el Smart Device como una aplicación Web Mobile, a través del Browser del dispositivo.

Pero si queremos que la aplicación sea nativa, interactuando con las funcionalidades del dispositivo (como la agenda de contactos, el calendario, el gps) y que tenga un look & feel similar al resto de las aplicaciones nativas del dispositivo, debemos encontrar otra solución.



Para llegar a ella, demos un rodeo. Supongamos que deseamos que algunos de los objetos de la aplicación que procesan y devuelven datos estructurados (esto es: transacciones, procedimientos y data providers), puedan ser consumidos por otros programas diferentes. Para ello, una buena alternativa si queremos un intercambio liviano, es, pensándolos como recursos (con lo que estaremos dentro de la arquitectura de diseño Rest), exponerlos como Web Services Rest.

De esta manera cualquier programa, conociendo la URL de cualquiera de ellos, podrá invocarlos a través del protocolo HTTP (con los métodos GET, POST, PUT y DELETE según corresponda). El servicio se ejecutará en el Server, accediendo a la base de datos y devolviendo como response al Cliente un archivo con la representación del recurso (en el formato JSON). El cliente deberá saber decodificar ese JSON para tomar las acciones que requiera.

Por ejemplo, si el invocado fue un data provider que en el json devuelve el listado de propiedades inmobiliarias, desplegarlo en la pantalla del Smart Device.

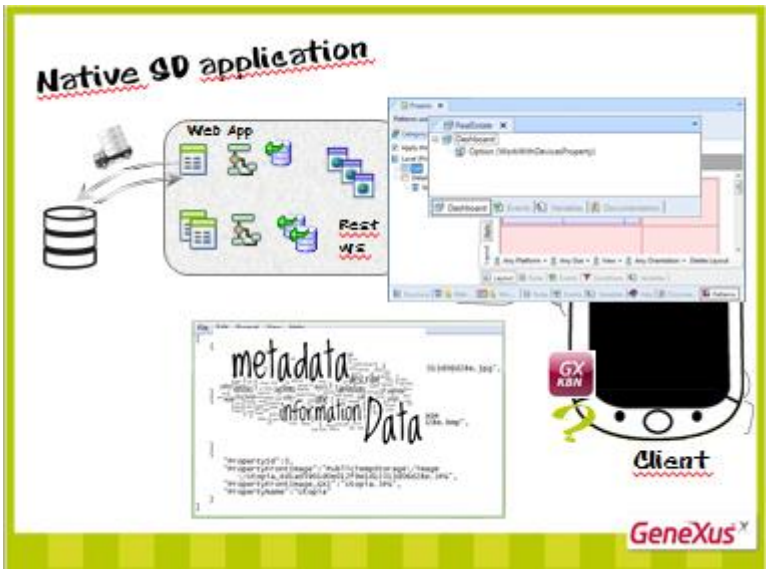
Por tanto, la recuperación y manipulación de datos en los dispositivos será a través de servicios Rest.

Demo

Recordemos que al aplicarle el pattern work with a una transacción…

…se creaba automáticamente el Business Component, exponiéndolo como servicio Rest.

Además, creará automáticamente data providers para recuperar la información del nodo List, y también del Detail y de cada Section del mismo, que también serán expuestos como servicios rest.



Ahora bien, no es un browser, entonces, ¿qué aplicación es la que invoca a estos servicios Rest desde el dispositivo? Y luego sabe decodificar los json recibidos para luego diseñar la pantalla que se despliega al usuario?

Y ¿dónde se encuentra la información de diseño del layout, que permita desplegar en pantalla la información extraída del json siguiendo ese diseño?

Tenemos dos opciones: ejecutar una especie de Browser especial creado por Artech, conocido como KBN o instalar la aplicación compilada. Estudiaremos ambas alternativas.

En ambos casos se mantendrá una metadata con la información de los objetos que componen la aplicación para Smart Devices, sus layouts, las URIs (Urls) de los servicios rest que deban ser invocados para recabar los datos, etcétera. En definitiva, la información de los patrones work with, así como de los dashboards y paneles para smart devices estarán encapsulados en esa metadata.

Empecemos por la primera opción nombrada: utilizar el KBN, Knowledge Base Navigator

Es un intérprete liviano instalado en el dispositivo, que se descarga del market place correspondiente a cada plataforma.

Tiene la lógica para:

- Leer la metadata, y las imágenes de la aplicación (que con esta solución estarán en el servidor web) utilizando arquitectura Rest (a través de los requests http correspondientes) y ejecutar los web services rest que sean necesarios, obteniendo los archivos Json con las respuestas con los datos, y a partir de la metadata leída y de los datos devueltos, armar la interfaz en el dispositivo.

- También es capaz de interpretar algunas acciones que el usuario dispara al operar con la aplicación (haciendo tap… o eligiendo alguna opción del menú) y producir nuevos requests al servidor para satisfacer las necesidades manifestadas por el usuario.

Por tanto el KBN permite a usuarios navegar a través de las aplicaciones para Smart Devices creadas con GeneXus. Dado que es como un Browser, que despliega las URLs correspondientes a objetos main de la aplicación, permitiendo así acceder a la metadata (presente en el server) y trabajar con las entidades y relaciones involucradas.

Por ejemplo, lee en la metadata la información del Dashboard de la URL elegida (las imágenes que debe utilizar -en este caso una, correspondiente a la única opción-, el texto de la opción, y a quién debe invocar al hacer tap sobre la misma y en base a esto arma y despliega la interfaz en el dispositivo.

Cuando el cliente hace tap sobre la opción, en la metadata, junto con toda la información de diseño, viene la URI para ejecutar el recurso: el data provider correspondiente al List de propiedades. Por lo que el KBN lo ejecuta vía Http rest y la respuesta es un JSON, con la lista de propiedades, que arma dentro de la interfaz cuyo diseño extrajo de la metadata en primera instancia.

Demo

Al hacer F5, como tenemos un único objeto para Smart Devices main –el dashboard RealEstate que habíamos creado- se abrirá en el emulador el Knowledge base navigator con una única URL para seleccionar: la correspondiente a ese dashboard.

Hagamos tap sobre la misma y veamos que se carga el dashboard con la única opción que tenía configurada.

La otra alternativa, que es la que utilizaremos cuando deseemos poner en producción, es encapsular la URL correspondiente al punto de entrada de la aplicación (por ejemplo, el dashboard), junto con toda la lógica del KBN y la metadata, en un archivo compilado en el lenguaje del dispositivo y descargar e instalar ese compilado. De esta forma no se requerirá del KBN, y tan sólo deberá accederse al server para ejecutar los Rest Web Services que manipulan los datos (dado que la metadata estará incluida en el paquete compilado e instalado en el propio dispositivo).

Cada plataforma de Smart Devices tiene su propio lenguaje y por tanto su propia extensión para el archivo compilado.

Ahora bien, ¿cómo obtenemos el compilado?

Indicando en las propiedades del generador SmartDevices cuál es el “Startup object”.

Haciendo esto, en el próximo F5 GeneXus entenderá que deberá compilar la aplicación. En particular para iOS se requerirá una Mac conectada en red con la pc de desarrollo. Lo veremos en otro video.

Demo

Así, configuremos para el generador de Smart Devices el Startup Object deseado: el dashboard RealEstate. F5

Como estamos prototipando en la nube en Android, al abrirse el Developer Menu Web, además de mostrarse los links para ejecutar la aplicación web (como de costumbre), vemos que está apareciendo el QR Code que está encapsulando la URL de este compilado.

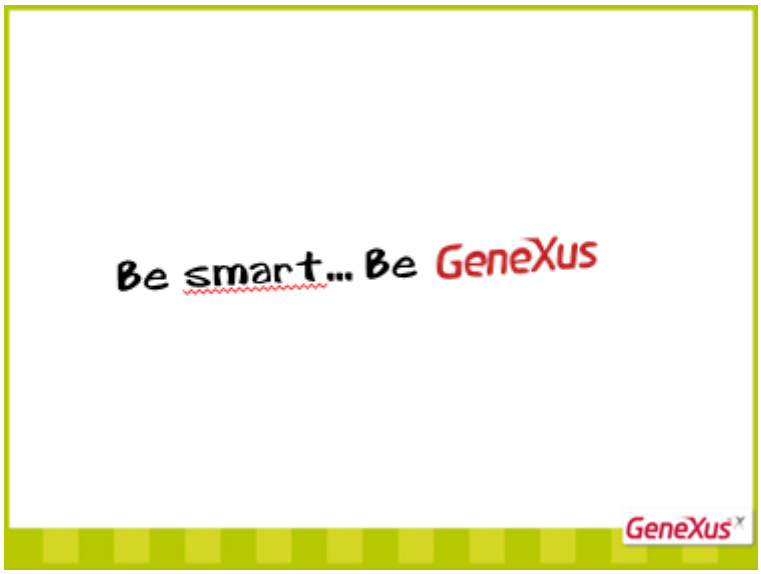
De esta manera, teniendo un dispositivo Android, a través del programa para lectura de QR codes que tenga instalado, se lo puede leer y descargar e instalar en el propio dispositivo.

Además, como podemos ver, en el emulador se está abriendo automáticamente el objeto Startup Object seleccionado, esto es, el dashboard RealEstate.

Debido a la arquitectura que acabamos de presentar con la utilización de los objetos como servicios rest, y en particular de las transacciones expuestas como business components, con lo cual estamos reutilizando todas las reglas del negocio, estamos haciendo uso de toda la potencia que nos brinda el servidor.

En otros videos abordaremos los detalles de la prototipación y la publicación y puesta en producción

Continuará…


Be smart... Be GeneXus
GeneXus