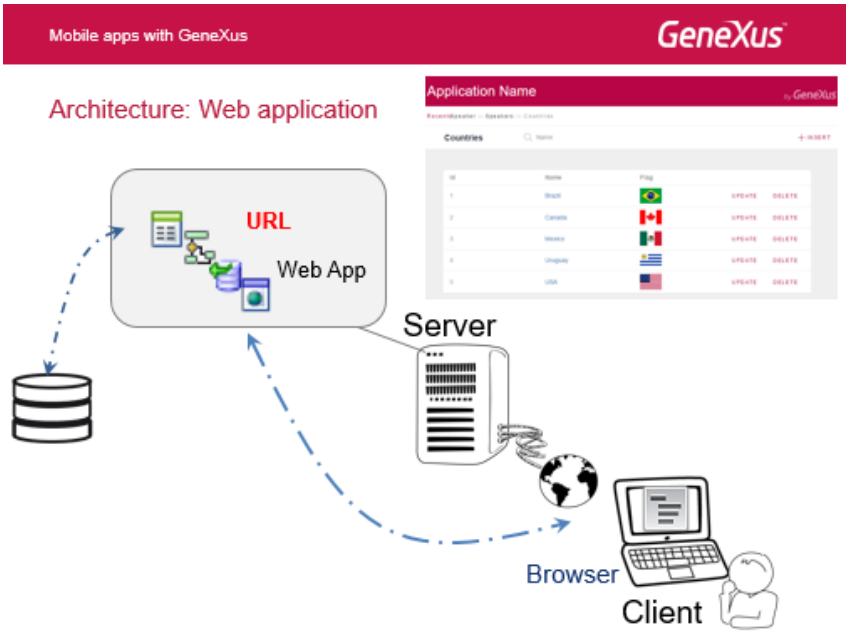
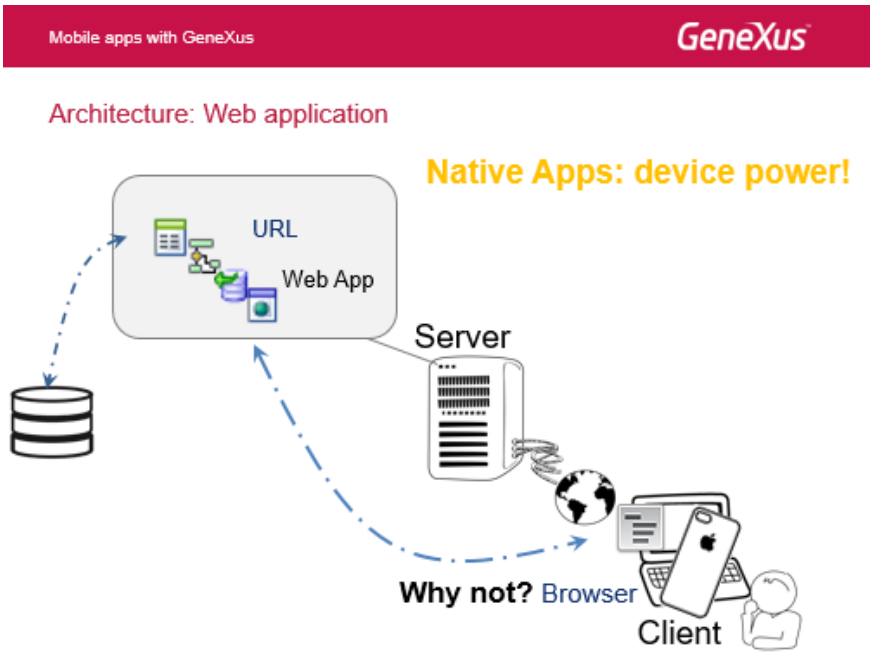
Online Arquitecture

Ahora vamos a enfocarnos en la arquitectura de las aplicaciones online y vamos a dejar la parte de aplicaciones offline para el final del curso

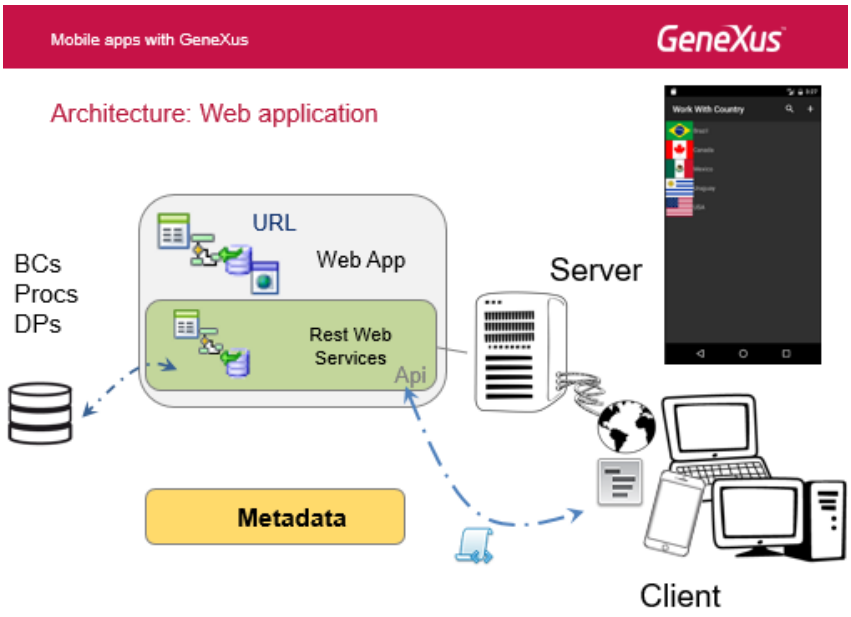


Para pensar la arquitectura subyacente en las soluciones para Smart Devices con GeneXus, partamos de lo conocido: las aplicaciones Web y hagamos la comparación que nos permitirá comprender un poco mejor.

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 que ejecuta, por ejemplo el WorkWithCountries. Este objeto web panel consulta la base de datos y devuelve la información al cliente, para que el Browser arme el layout que presentará al usuario como un HTML como respuesta a su pedido.

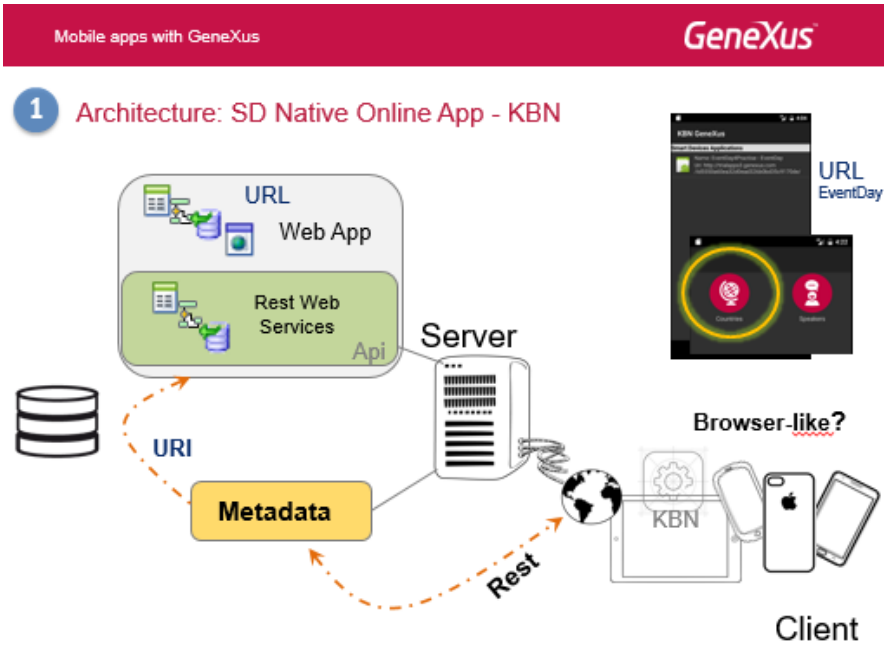


Si queremos la aplicación ejecutándose en un Smart Device, ¿por qué implementar una solución particular en vez de usar la web, a través del navegador del propio dispositivo?
Es que queremos que la aplicación interactúe con las funcionalidades propias del dispositivo, como la agenda de contactos, el calendario, y demás, y que tenga un look & feel similar al resto de las aplicaciones nativas, como habíamos visto antes.



Seguimos entonces exclusivamente con la aplicación web. Generalizando: si deseamos que algunos de los objetos de la aplicación que procesan y devuelven datos estructurados por ejemplo transacciones como business components, los procedimientos o los data providers, si queremos enctonces que esos objetos puedan ser consumidos por otros programas (no necesariamente implementados por GeneXus) a través de Internet (tanto desde una notebook o pc, como de un smart device), una buena alternativa es exponerlos como **Rest Web Services** (serán, así, apis de la aplicación, conformando una **capa de servicios**). Con ello nos encontramos dentro de una arquitectura de diseño Rest, que piensa en esos programas como **recursos**.

De esta manera cualquier programa que acceda a internet, conociendo la URL de cualquiera de estos **web services**, podrá invocarlos a través del protocolo HTTP (con los métodos GET, POST, PUT, DELETE según corresponda). El **servicio** se ejecutará en el Server, accediendo a la base de datos (por ejemplo podemos pensar en un data provider que devuelva una colección de banderas y nombre de los países) y devolviendo entonces como respuesta, lo ejecuta, el Cliente invoca a esa URL a ese servicio el data provide lo ejecuta y lo que hace es devolver al cliente un JSON con esa colección entonces de paises. El cliente deberá saber decodificar ese JSON, para hacer con su información lo que necesite.
Así, si el cliente se ejecuta en un dispositivo inteligente, y éste accede a una metadata (probablemente en el propio servidor) que contiene toda la información para armar la interfaz del work with (entre otras cosas), sabrá entonces cómo armar el layout del Work With Country, a partir del Json recibido con los datos, va a poder mostrar la lista en el dispositivo. Por allí vendrá la solución que estamos buscando.



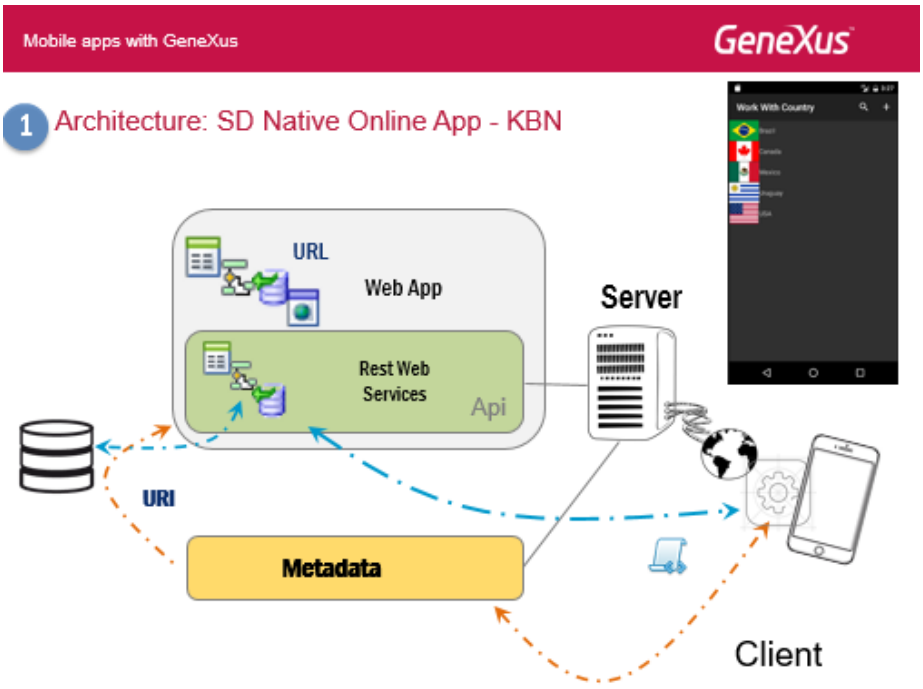
La **metadata** de la aplicación para Smart Devices contendrá entonces toda la información de la aplicación (qué dashboards, work with for smart devices y panels implementa, y también las URIs de los web services para obtener los datos necesarios de la base de datos) para poder armar así la interfaz de la aplicación en el dispositivo y responder a las acciones que el usuario ejectua en ese dispositivo. Estará la metadata en el servidor web.

Tenemos dos opciones para prototipar: ejecutar una especie de navegador especial creado por GeneXus, el KBN, o instalar la aplicación compilada. Vamos a estudiaremos ambas soluciones.

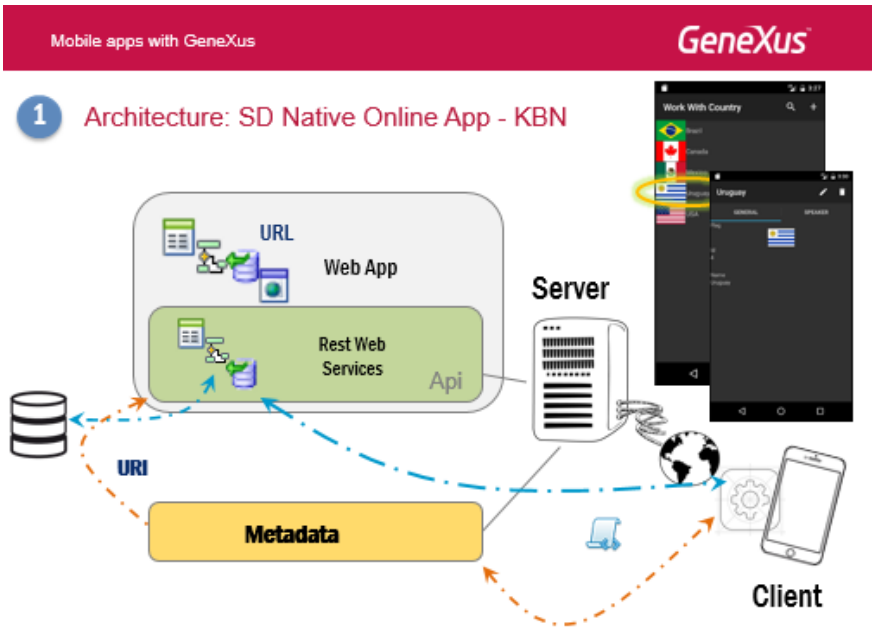
Vamos a empezar por el KBN

- Es una aplicación nativa (compilada en el lenguaje de la plataforma) creada por GeneXus.
- Permite navegar a través de las aplicaciones para Smart Devices creadas con GeneXus como si fuera un Browser, eligiendo una URL (correspondiente a un objeto main de la aplicación, consignado en la metadata), va a permite de esa manera trabajar con las entidades y relaciones que conforman la parte de la aplicación GeneXus para Smart Devices que depende de ese main.
- Es un **intérprete** liviano, que tiene la lógica para leer la metadata de la URL correspondiente, así como las imágenes de la aplicación y va a poder decodificarla toda esa información, invocando, de ser necesario, a los web services rest que necesite para obtener las respuestas con los datos, para armar la interfaz correspondiente en el dispositivo, que es la que visualiza el usuario.

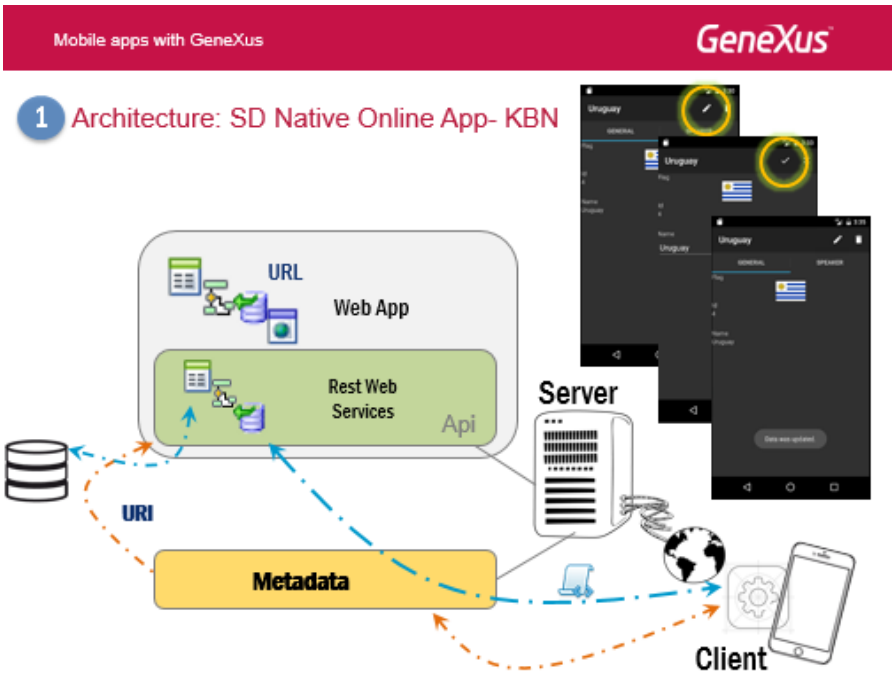
Por ejemplo, lee en la metadata que debe comenzar por el **dashboard** EventDay, que tiene tales imágenes, y tal interfaz (layout) y tales opciones. Arma la interfaz y la despliega en el dispositivo. Cuando el cliente hace tap sobre la opción, de la metadata obtiene la URI o URL para ejecutar el recurso: en el ejemplo, Countries (un data provider que devuelve la lista de países, consultando la base de datos).



Por lo que el KBN lo ejecuta vía Http (rest), con lo cual accede a la base de datos para obtener la colección de países en un JSON, y este Json se le devuelve como respuesta al KBN, que habiendo accedido además a la metadata, tiene todo lo que necesita para armar la pantalla que se muestra al usuario en el dispositivo.

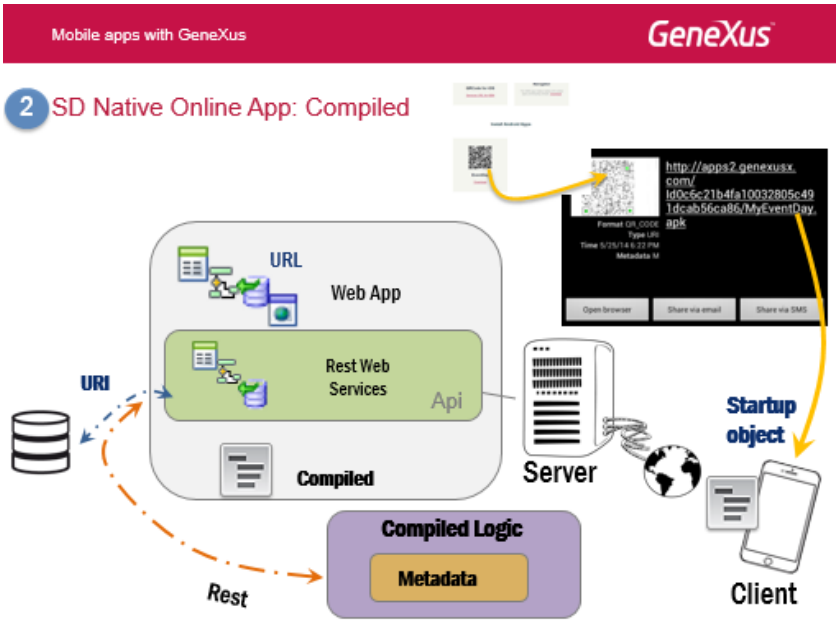


Análogamente, si se hace tap sobre un elemento de la lista (por ejemplo Uruguay) se llamará al data provider que devuelve la información del país, para armar la pantalla del View.



Luego, si se hace un update o un delete (o un insert desde el list), se dibuja la pantalla de Edit, y al intentar grabar, el servicio rest invocado será el Business Component, que intentará realizar la operación correspondiente sobre la base de datos, y devolverá al llamador el resultado de la operación (si falló, los mensajes de error ocurridos, para que se le muestren al usuario en la pantalla del dispositivo; si fue exitoso, un mensaje indicándolo).

Esa es la primera opción, utilizar el KBN, que como decíamos no era la opción más indicada sobre todo pensando en Androi, porque el KBN al ser un intérprete no nos va a permitir interactuar ni con todas las funcionalidades del dispositivo y además no tendrá incorporadas todas las funcionalidades que una aplicación compilada si tendrá.



Opción 2: Compilar la aplicación en el lenguaje del dispositivo, e instalarla en el mismo.
Cada plataforma de Smart Devices tiene su propio lenguaje y por tanto su propia extensión para el archivo compilado. Por ejemplo, para Android es .apk. Este archivo debe descargarse e instalarse en el dispositivo, y ya no se va a necesitar del intérprete KBN. Encapsulará toda la metadata y las imágenes.
Entonces lo vamos a descargar e instalar en el dispositivo, ese compilado va a contener toda la lógica y la metadata y solamente va a necesesitarse accede al servidor para ejecutar los servicios Rest que devuelve los datos de la base de datos.
Para compilar la aplicación para Android, alcanza con indicar en las propiedades del Environment (e.g. C# Environment) el "**Startup object**" (por ejemplo, el dashboard EventDay). En este caso al hacer F5 no se va generar más la aplicación web, y por tanto no se va a abrir el browser ni se va a mostrar el libre por menu, sino que únicamente se va a compilar ese objeto, el dashboard y todos los objetos que dependan de él, y ese compilado es el que va a quedar en el servidor, desde ese lugar es que lo vamos a poder descargar para instalar en el dispositivo, o si estamos con en el emulador se va a abrir directamente en el emulador.