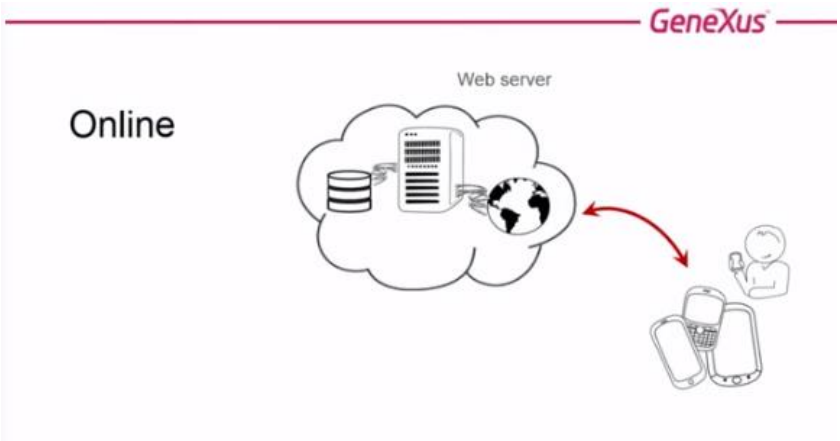
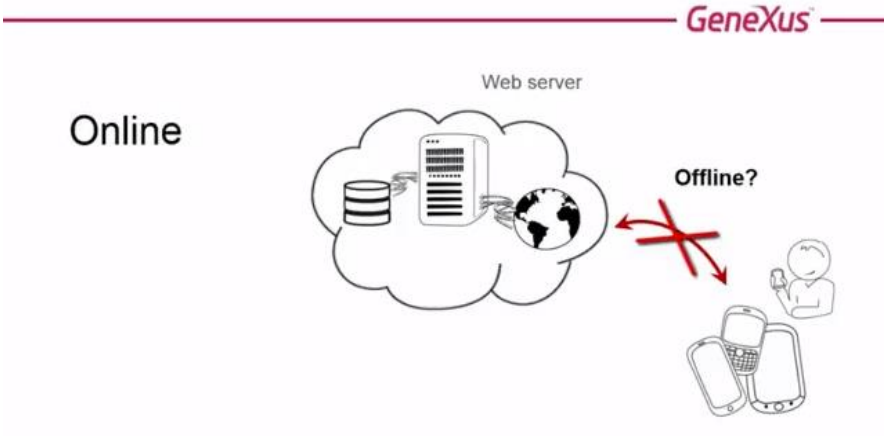
Arquitectura de aplicaciones móviles online

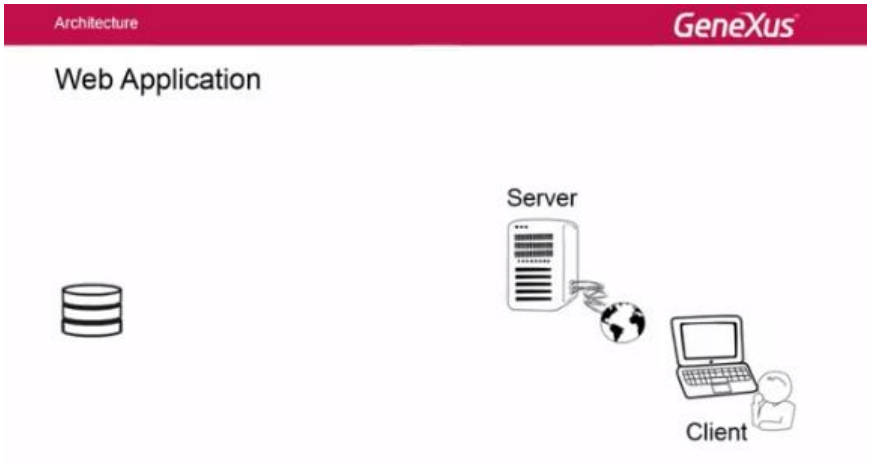
En este video nos enfocaremos en la arquitectura de las aplicaciones **online:**



y dejaremos el tratamiento de las aplicaciones **offline** para más adelante:



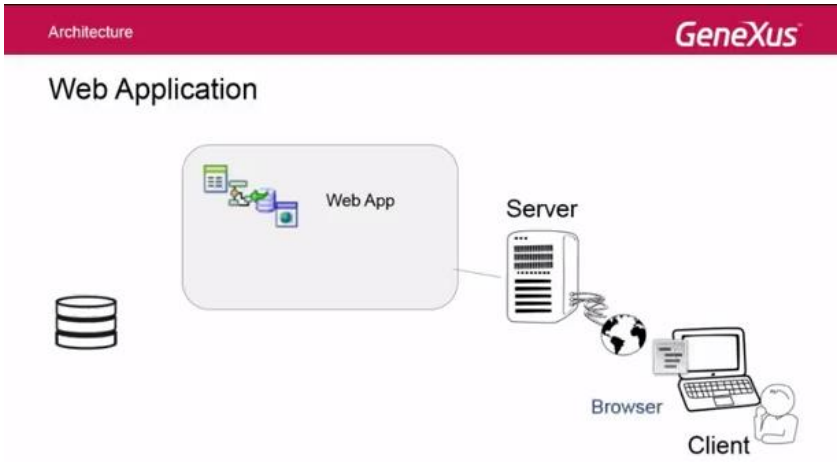
De hecho, como la prototipación de las aplicaciones offline requiere necesariamente que la aplicación se compile cada vez, y esto es más lento que usar el intérprete, como veremos, seguramente usted optará por hacer las primeras pruebas de la aplicación online, y dejará la prototipación offline para el final.



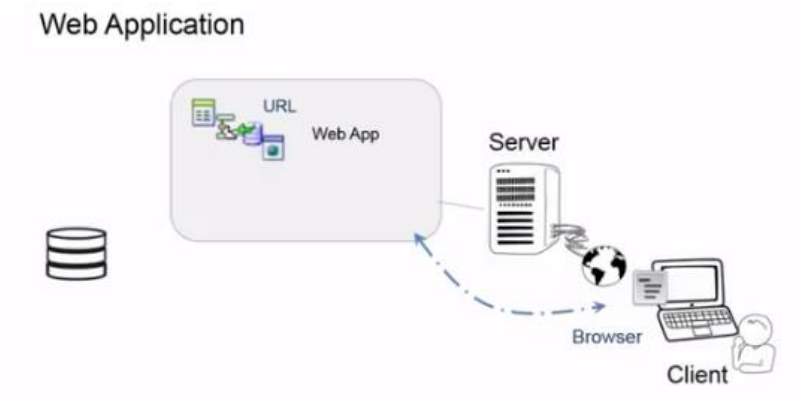
Para pensar la arquitectura subyacente en las soluciones para Smart Devices con GeneXus, partamos de lo conocido: las aplicaciones Web.

Tenemos por un lado un Servidor y por otro un Cliente.

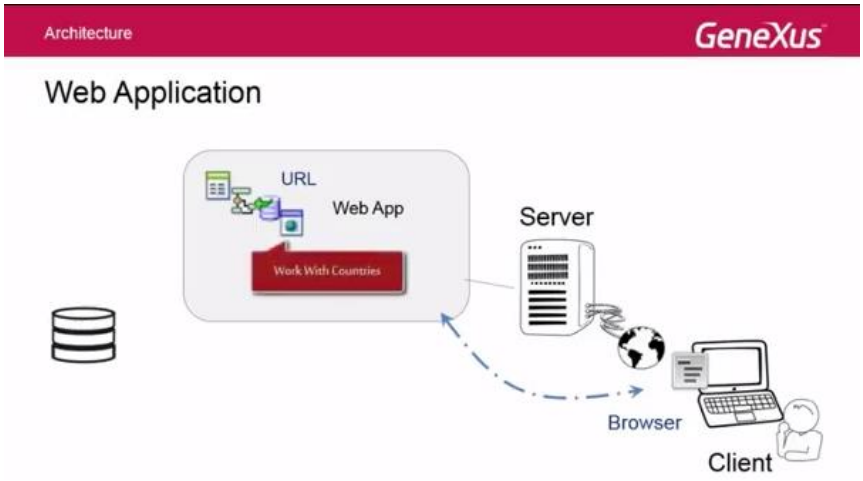
En el servidor tenemos la aplicación web y en el cliente un Browser:



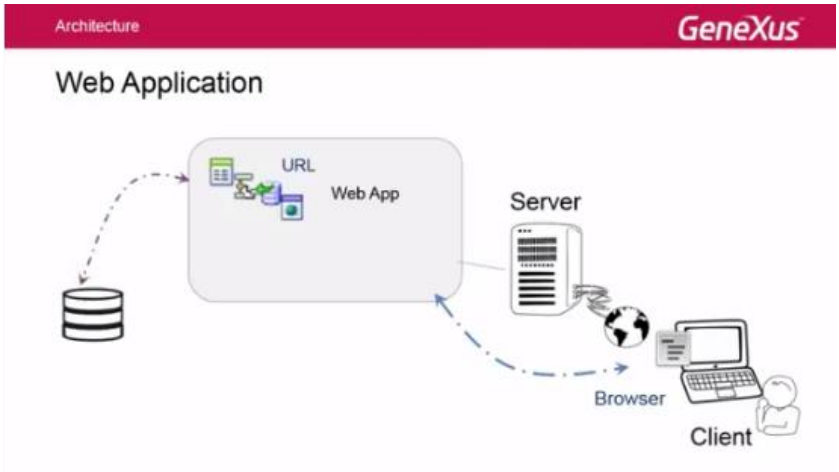
Ejecutamos la aplicación Web a partir de una URL:



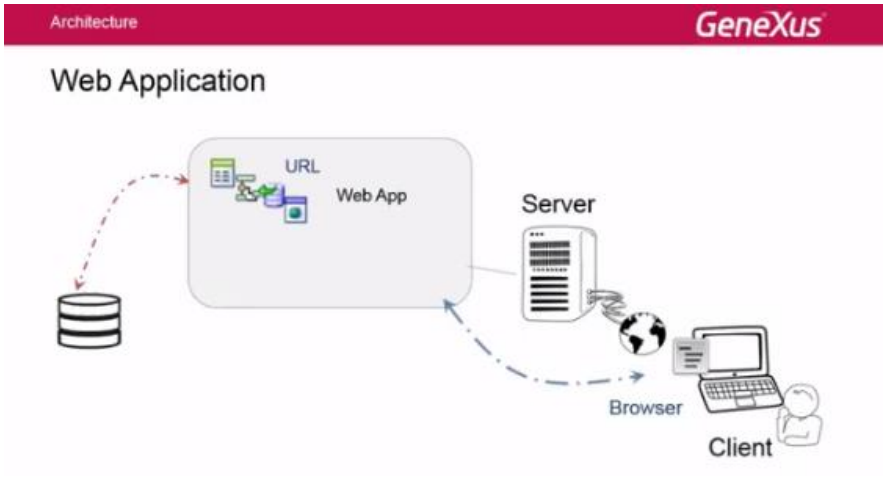
que ejecuta, por ejemplo el WorkWithCountries:



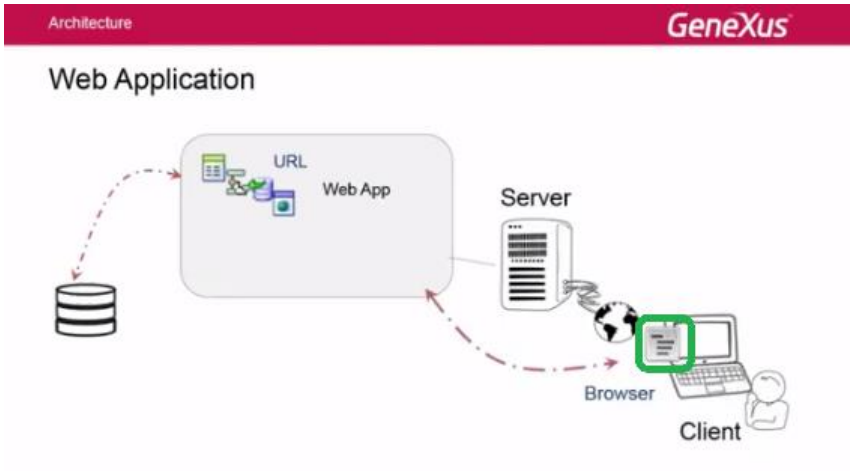
Este objeto consulta la base de datos:



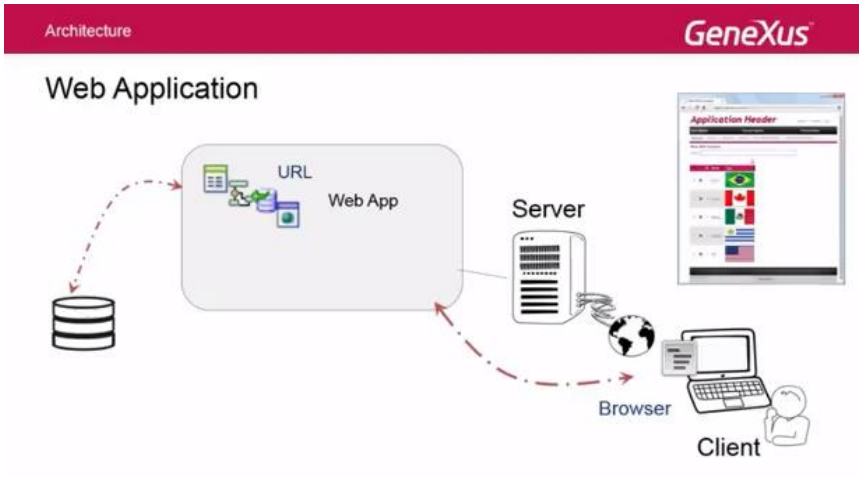
y devuelve la información al cliente



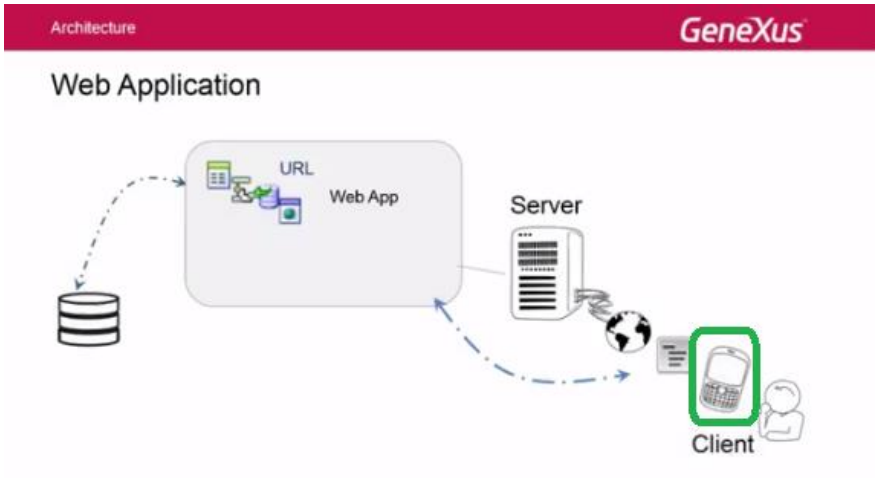
para que el Browser arme el layout



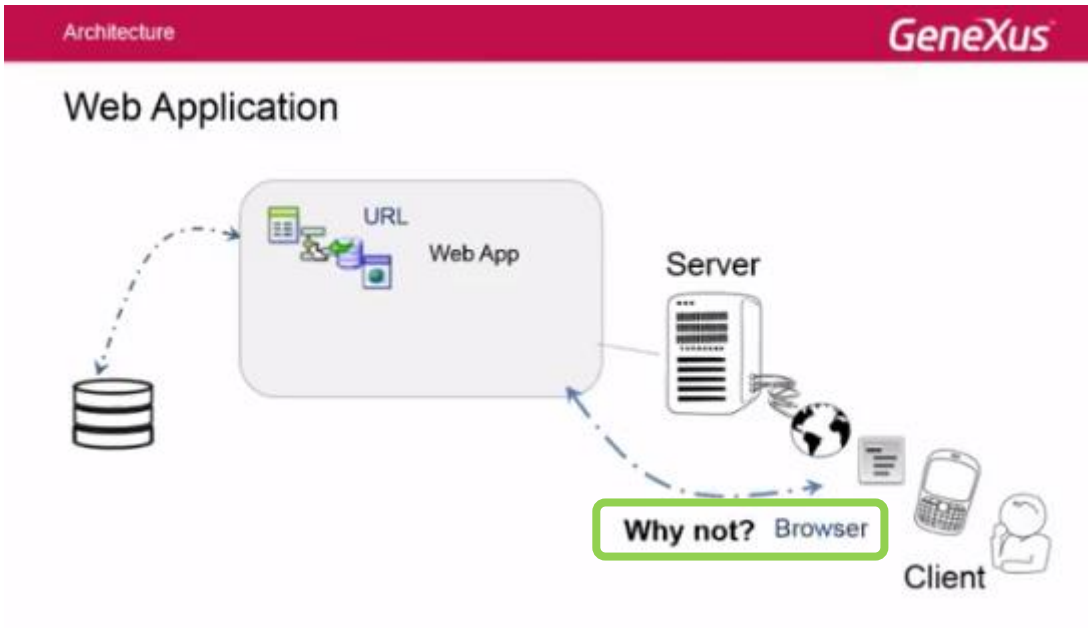
que presentará al usuario como respuesta a su pedido:



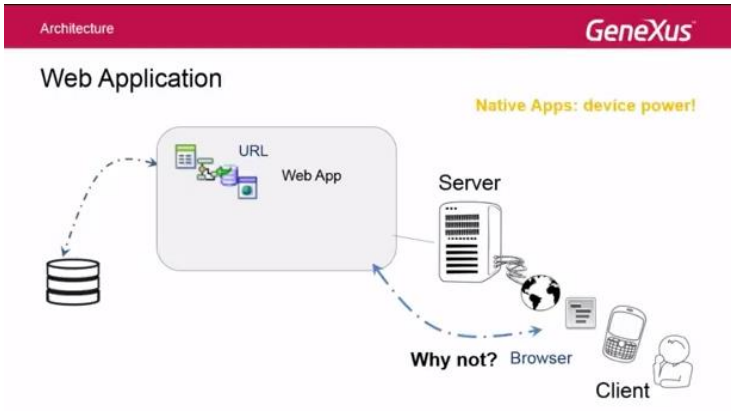
Si queremos la aplicación ejecutándose en un Smart Device:



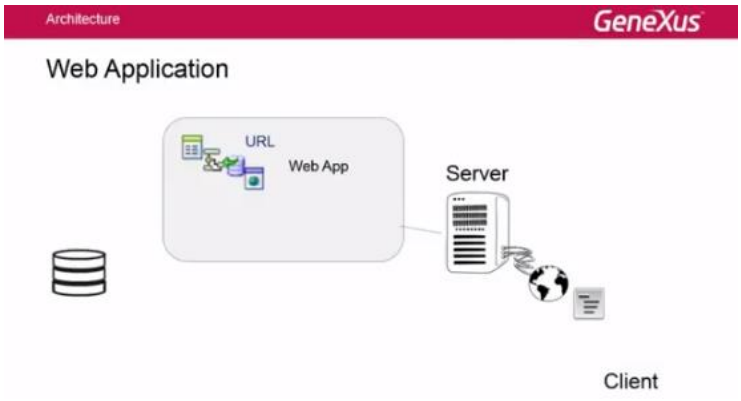
¿por qué implementar una solución particular en vez de usar la web, a través del navegador del dispositivo?



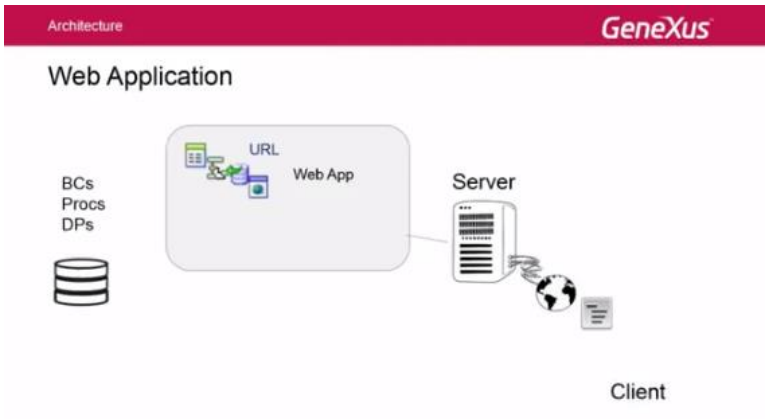
Es que queremos que la aplicación interactúe con las funcionalidades propias del dispositivo, como la agenda de contactos, el calendario, el GPS, y demás.. y que tenga un look & feel similar al resto de las aplicaciones nativas:



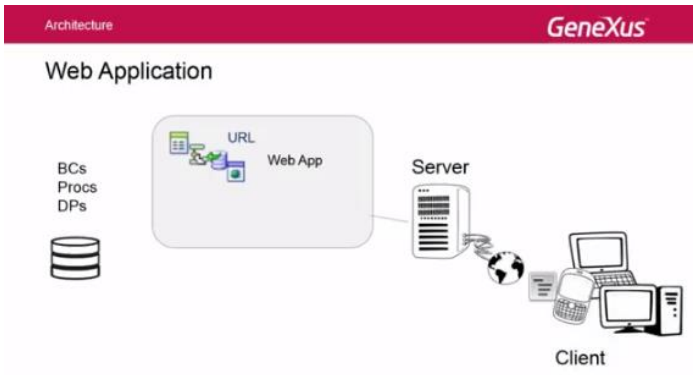
Sigamos entonces pensando exclusivamente en la aplicación web:



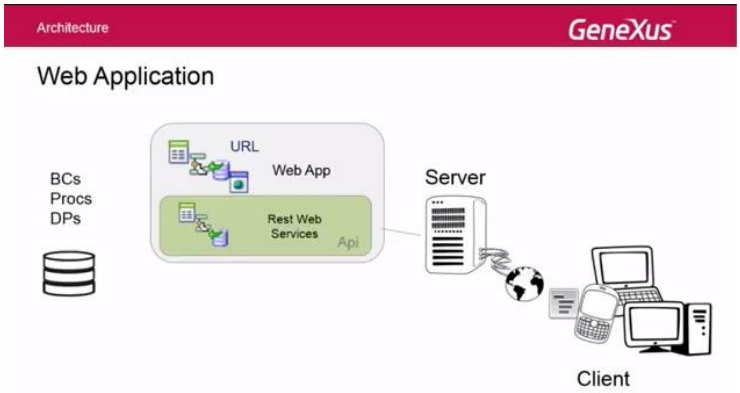
Generalizando: Si deseamos que algunos de los objetos de la aplicación que procesan y devuelven datos estrucurados (esto es: transacciones como business components, procedimientos y data providers)..



Si deseamos entonces que algunos de esos objetos puedan ser consumidos por otros programas (no necesariamente implementados con GeneXus) **a través de internet** (tanto desde una notebook o pc, como de un smart device):



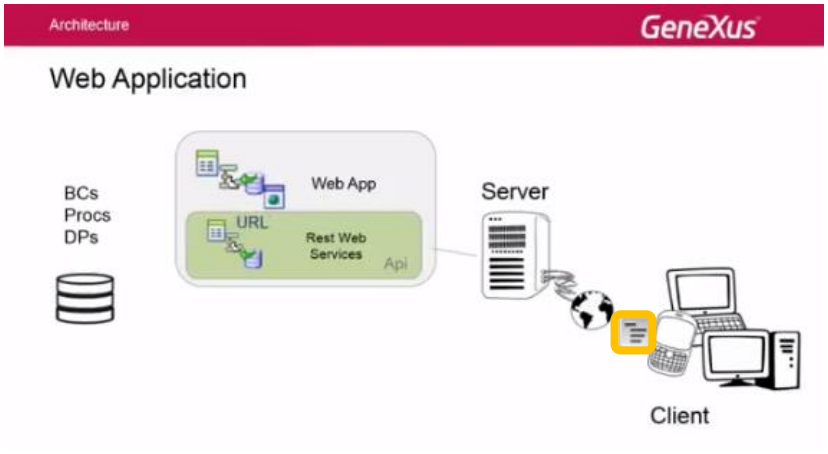
una buena alternativa es exponerlos como **Rest Web Services**



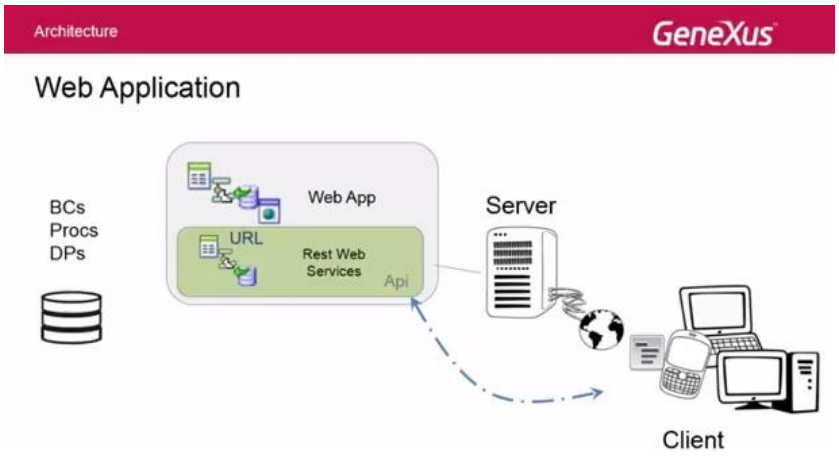
(serán, así, Apis de la aplicación, componiendo una **capa de servicios**).

Con ello nos encontramos dentro de una arquitectura de diseño Rest, **que piensa en esos programas como recursos.**

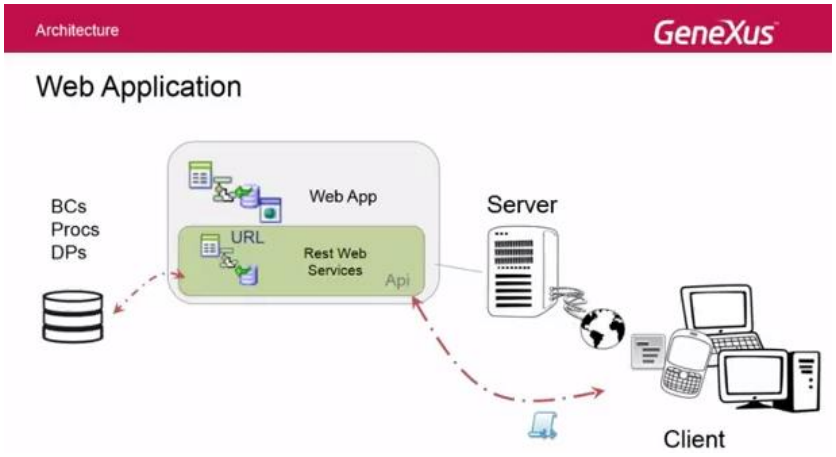
De esta manera cualquier programa que acceda a internet, conociendo la URL de cualquiera de estos **web services**



podrá invocarlos a través del protocolo HTTP (con los métodos GET, POST, PUT, DELETE según corresponda).

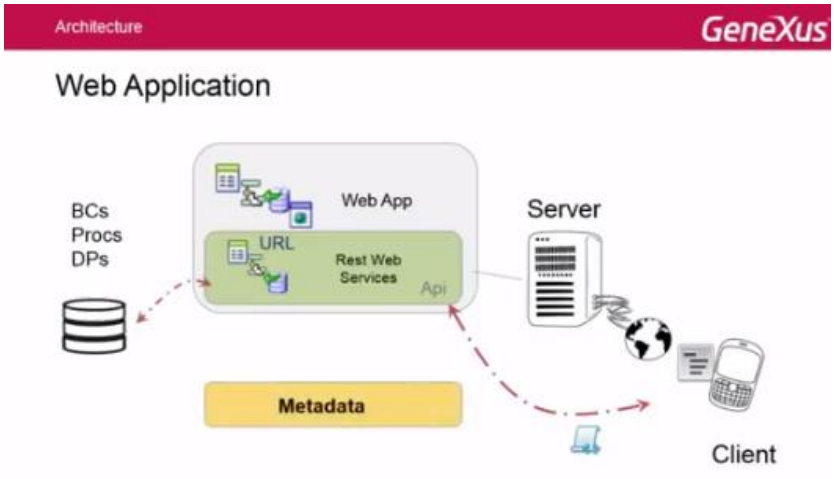


El **servicio** se ejecutará en el Server, accediendo a la base de datos (por ejemplo un data provider cuya salida sea una colección de "bandera y nombre" de los países) y devolviendo como response al Cliente la representación del recurso (con el formato por ejemplo JSON).

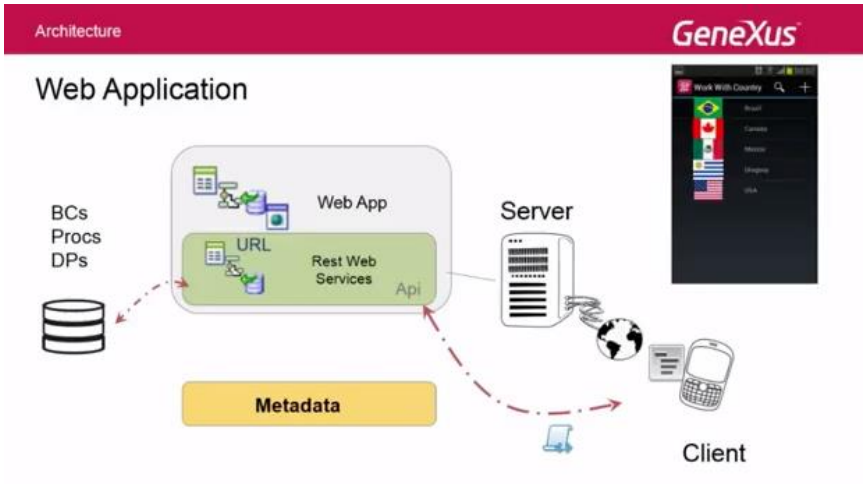


El cliente deberá saber decodificar ese JSON, para hacer con su información lo que necesite.

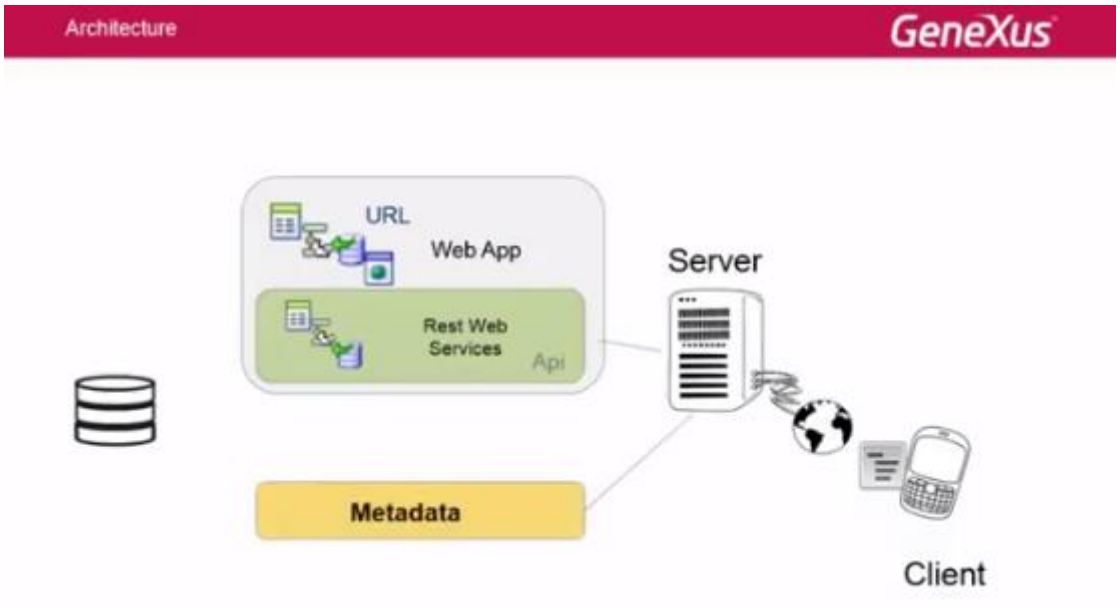
Así, si el cliente se ejecuta en un dispositivo inteligente, y éste accede a una metadata (probablemente en el propio servidor)



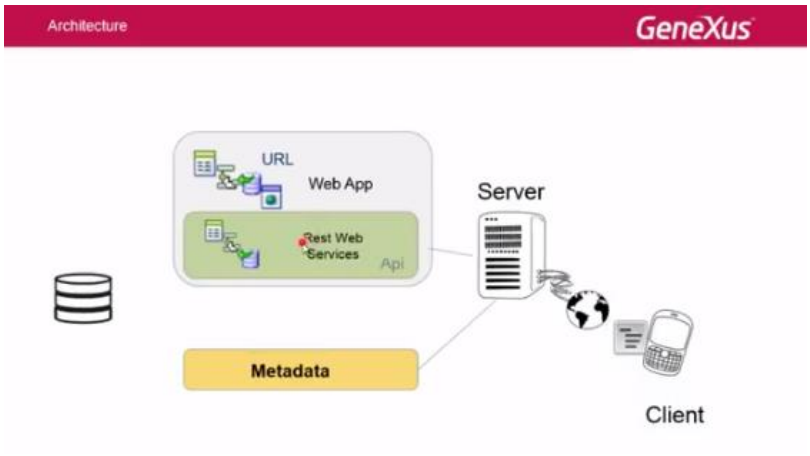
que contiene la información para armar la interfaz del work with (entre otras cosas), sabrá cómo armar el layout del Work With Country, a partir del Json recibido con los datos, mostrando la lista en el dispositivo:



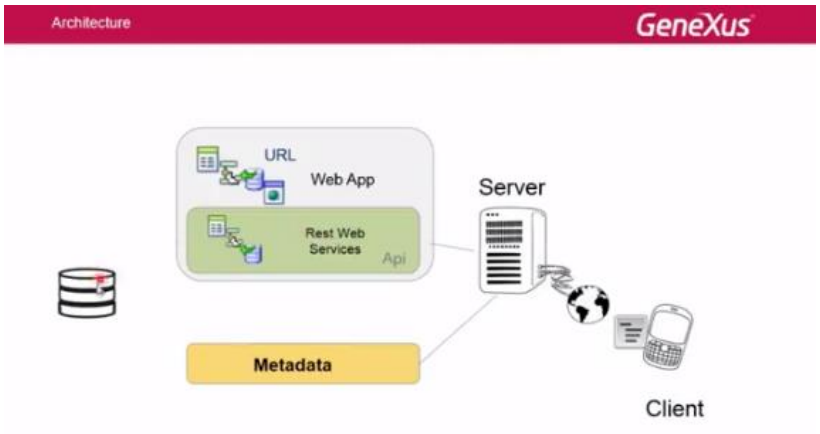
Por allí vendrá la solución online que estamos buscando.



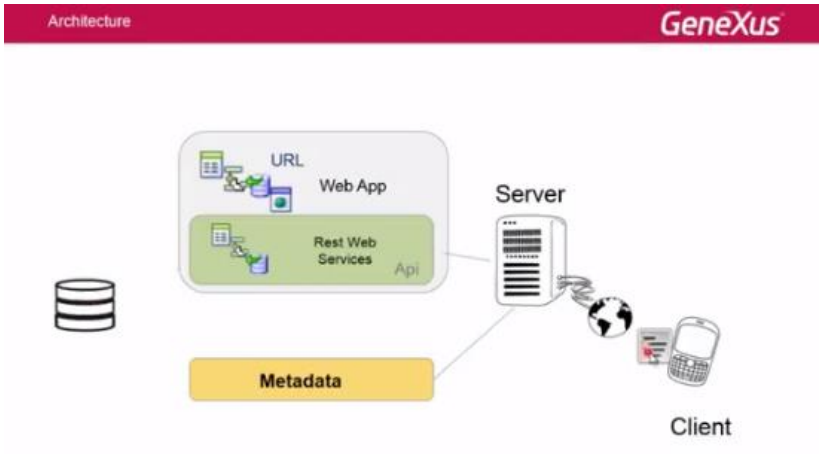
La **metadata** de la aplicación para Smart Devices contendrá toda la información de la aplicación: qué dashboards, work with smart devices y panels implementa, y las URis de los web services



para obtener los datos necesarios de la base de datos



para poder armar la interfaz de la aplicación en el dispositivo

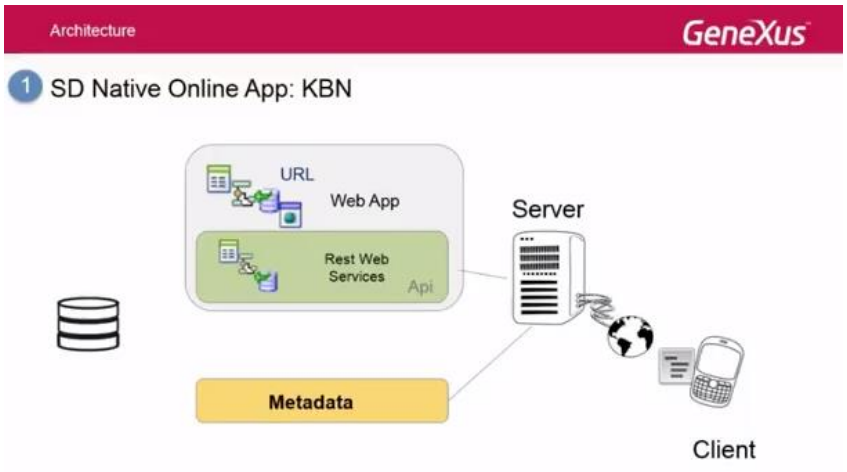


y responder a las acciones que se ejecutan.

Estará en el servidor web.

Tenemos dos opciones para prototipar:

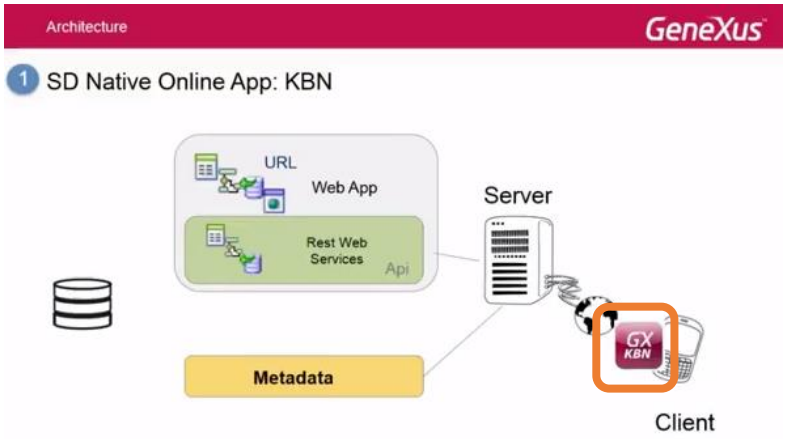
Ejecutar una especie de navegador especial creado por Artech, el KBN:



O instalar la aplicación compilada.

Estudiaremos ambas.

El KBN:

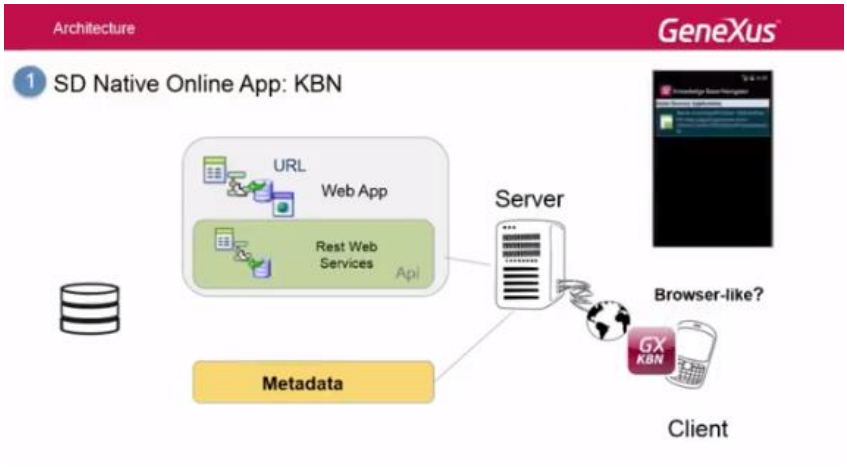


es una aplicación nativa (compilada en el lenguaje de la plataforma) que debe instalarse en el dispositivo antes de empezar a prototipar.

Se descarga del market place del dispositivo.

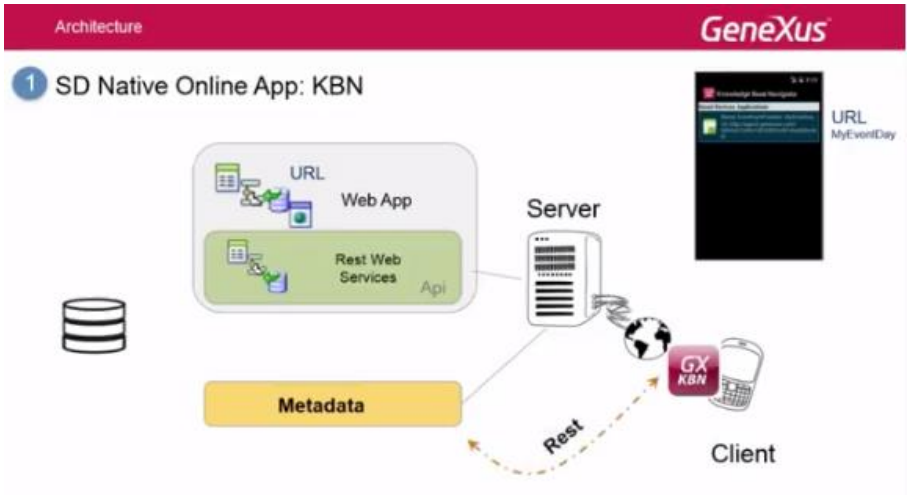
Permite navegar a través de las aplicaciones para Smart Devices creadas con GeneXus.

Es como un Browser:

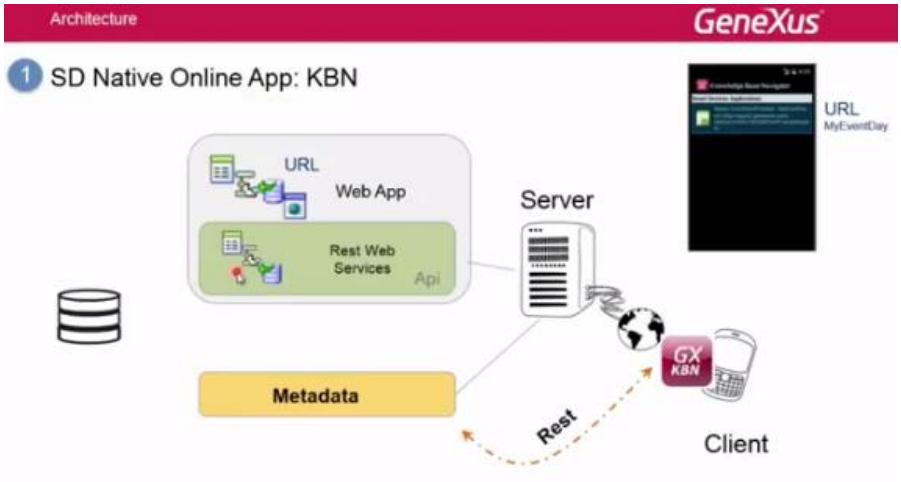


que eligiendo una URL (correspondiente a un objeto main de la aplicación, consignado en la metadata), permite trabajar con las entidades y relaciones que conforman la parte de la aplicación GeneXus para Smart Devices que depende de ese main.

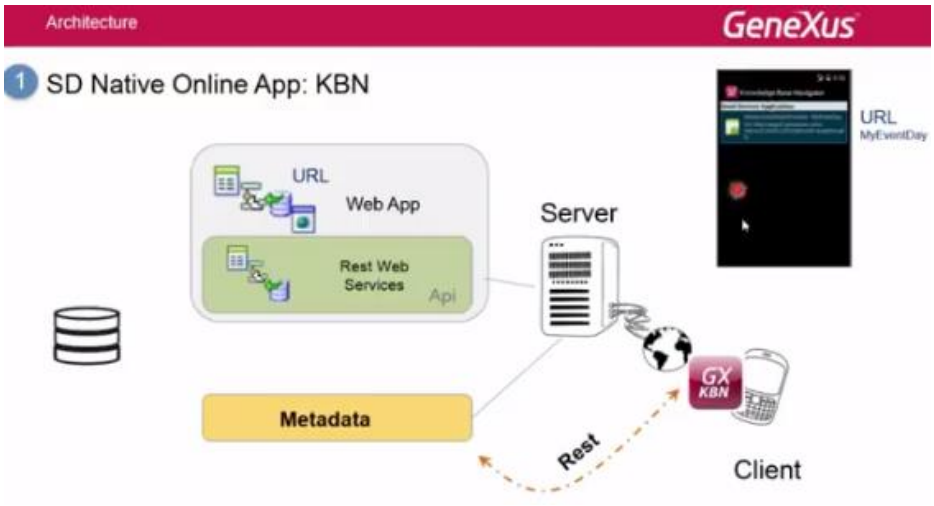
Es decir, es un intérprete liviano, que tiene la lógica para **leer** la metadata



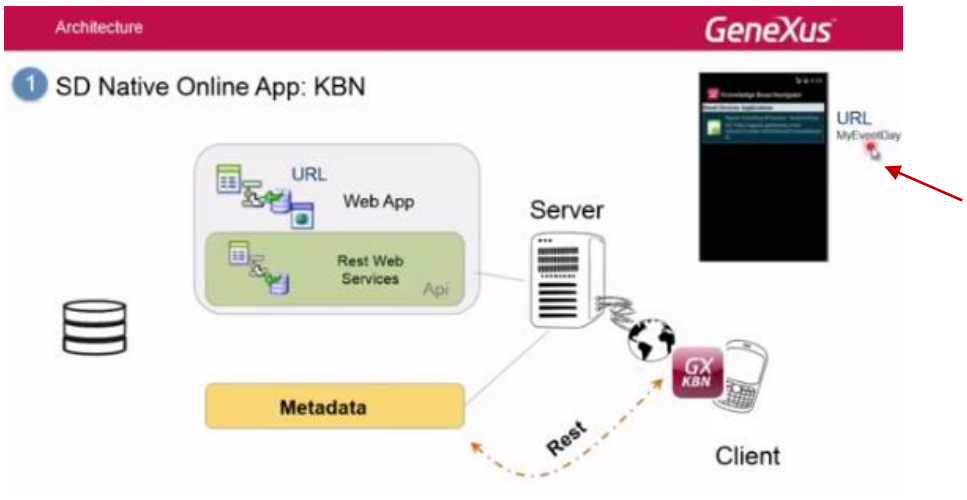
de la URL correspondiente, así como las imágenes de la aplicación y decodificarla, invocando, de ser necesario, a los rest web services



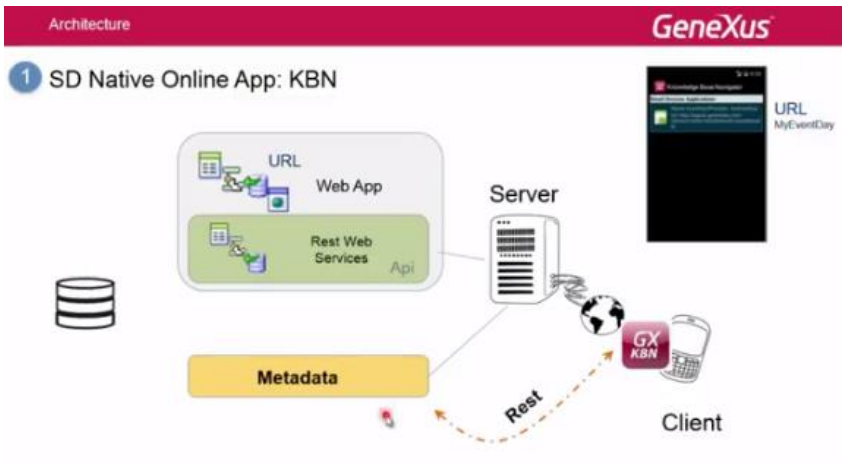
que requiera para obtener las respuestas con los datos, para armar la interfaz correspondiente en el dispositivo, que es la que visualiza el usuario.



Por ejemplo: lee en la metadata

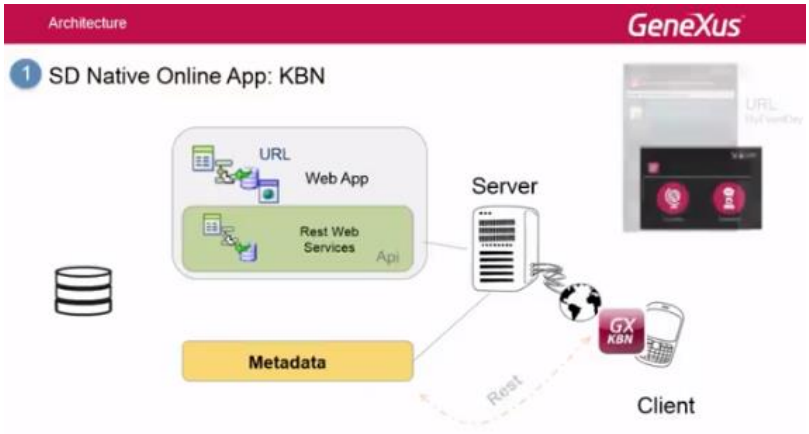


que debe comenzar por el **dashboard** MyEventDay



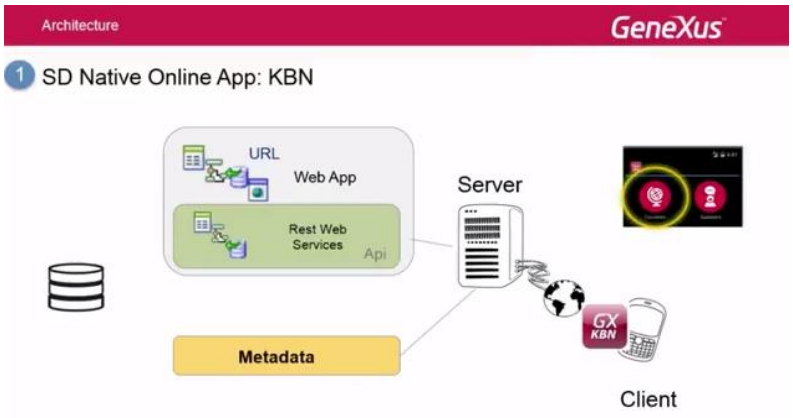
que tiene tales imágenes, y tal interfaz y tales opciones.

En este caso como el Dashboard no requiere ninguna consulta a la base de datos, arma la interfaz

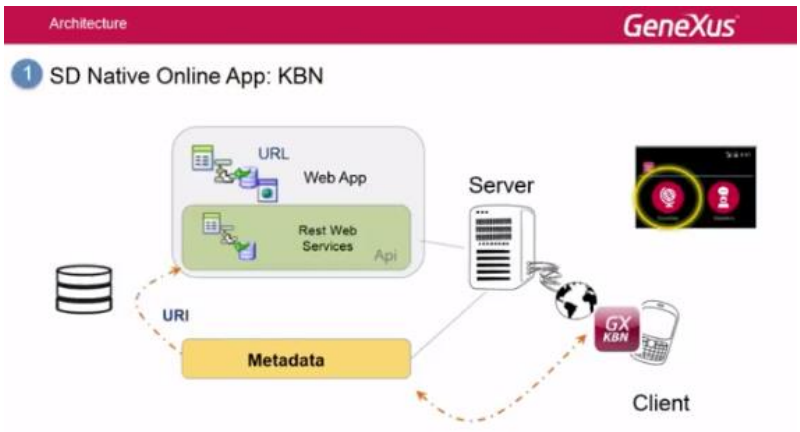


y la despliega en el dispositivo.

Cuando el cliente hace tap sobre la opción…

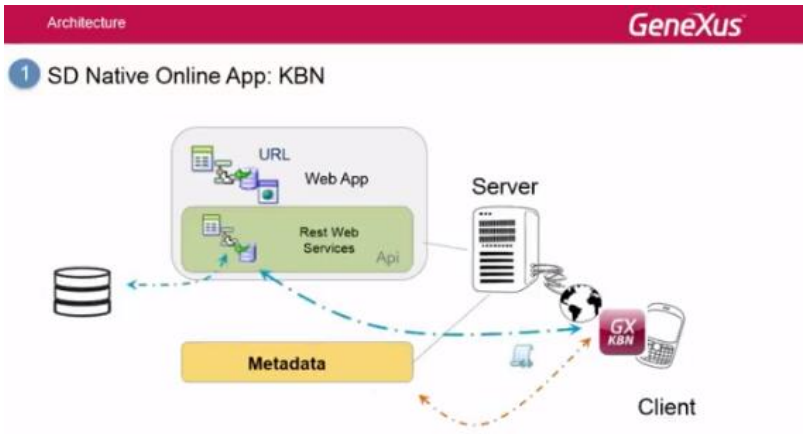


de la metadata obtiene la URI

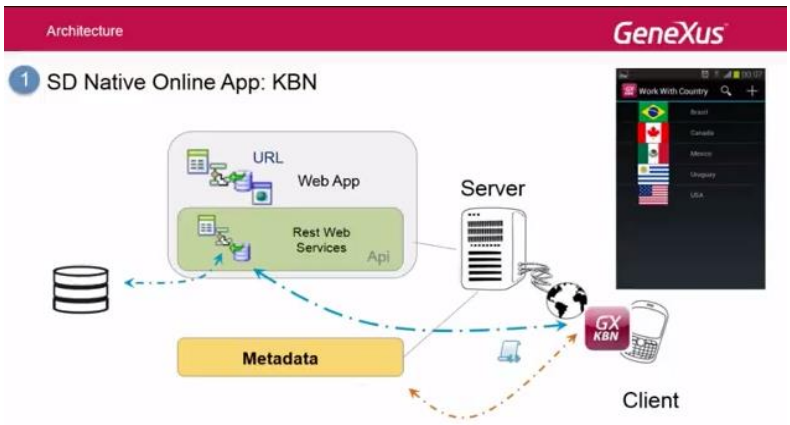


para ejecutar el recurso: en el ejemplo “Countries” (un data provider que devuelve la lista de países).

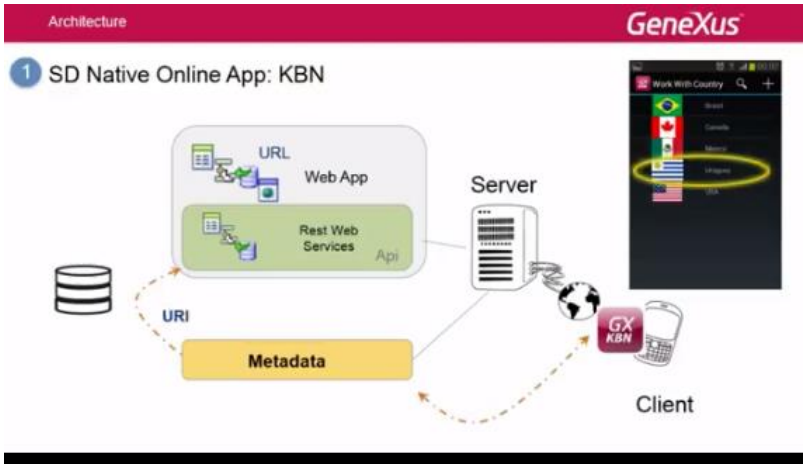
Por lo que el KBN lo ejecuta vía Http (rest), con lo cual el Data Provider accede a la base de datos para obtener la colección de países en un Json



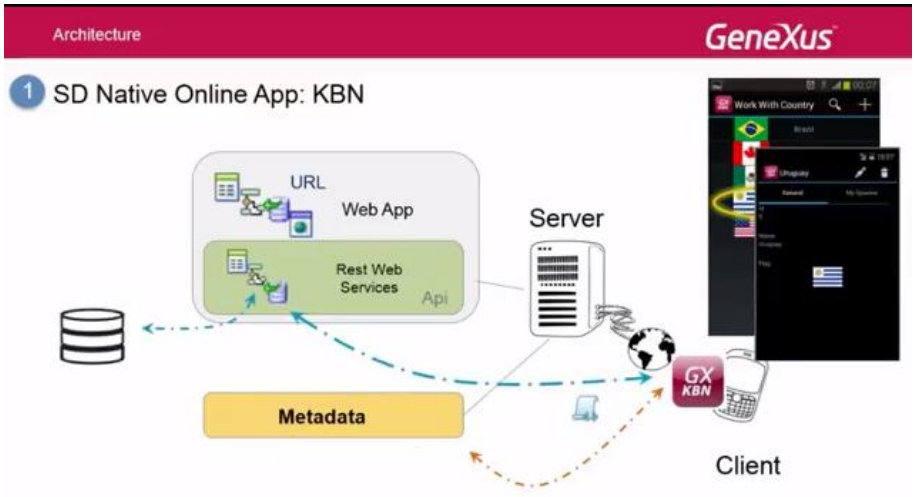
y este Json se le devuelve como respuesta al KBN, que habiendo accedido además a la metadata, tiene todo lo que necesita para armar la pantalla que se muestra al usuario en el dispositivo:



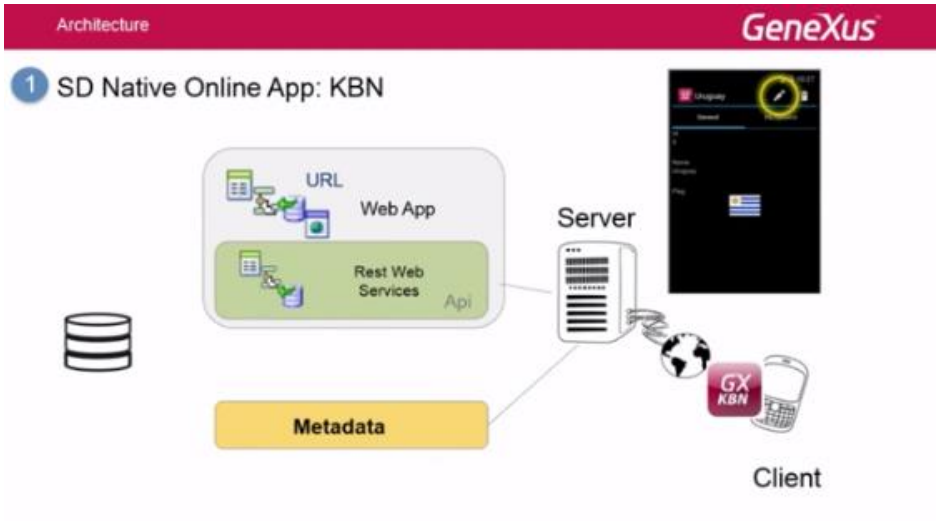
Análogamente, si se hace tap sobre un elemento de la lista



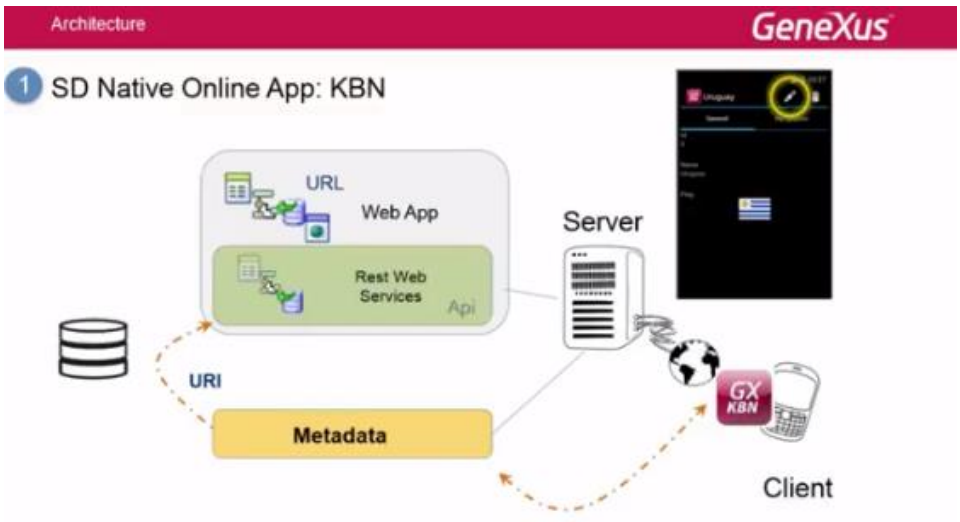
se llamará al Data Provider que devuelve la información del país, para armar la pantalla del View.



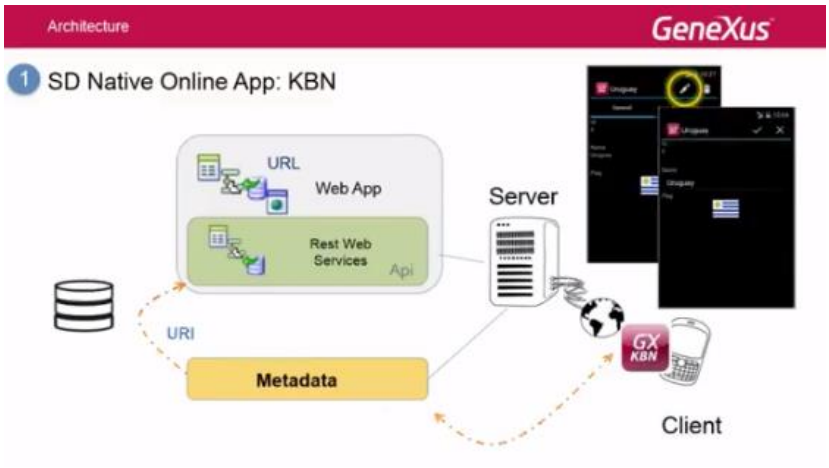
Luego, si se hace un update



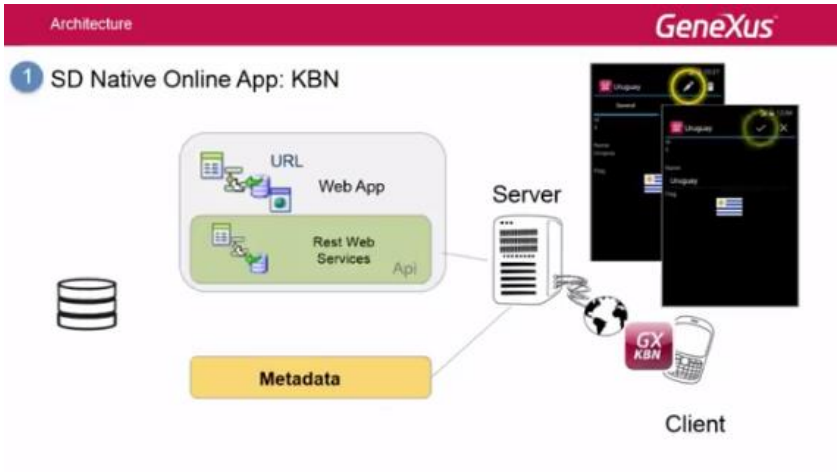
o un delete (o un insert desde el list), se encuentra que debe llamarse



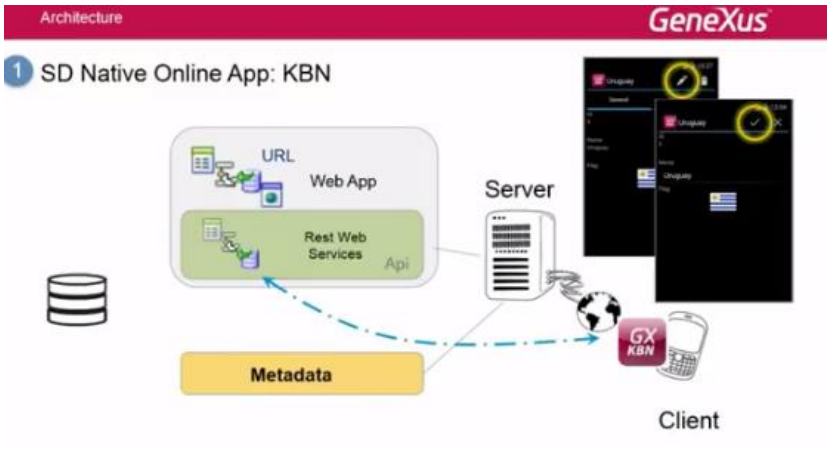
a la pantalla de Edit



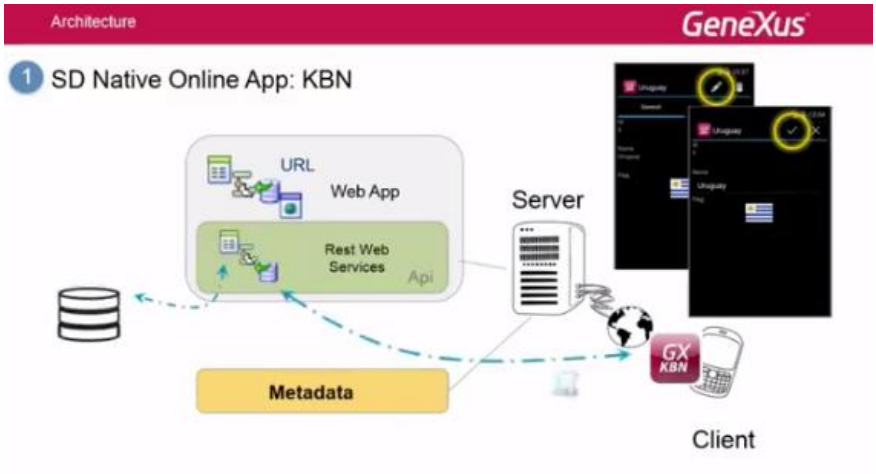
y al grabar



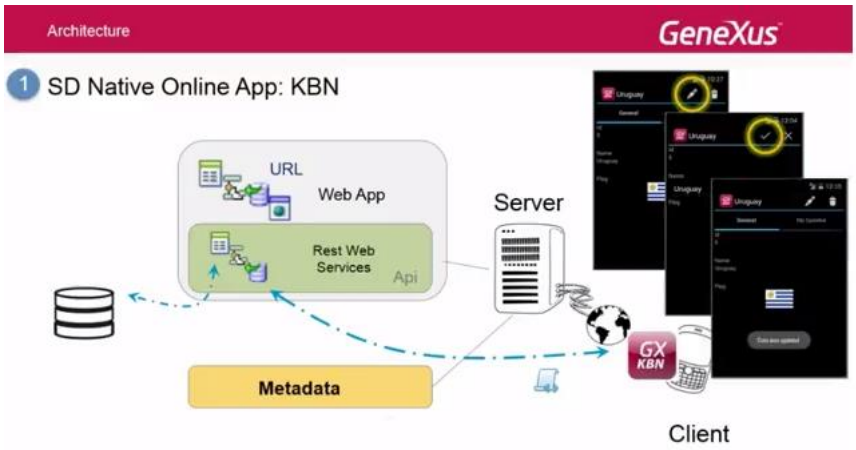
el servicio Rest invocado será el Business Component



que intentará realizar la operación correspondiente sobre la base de datos



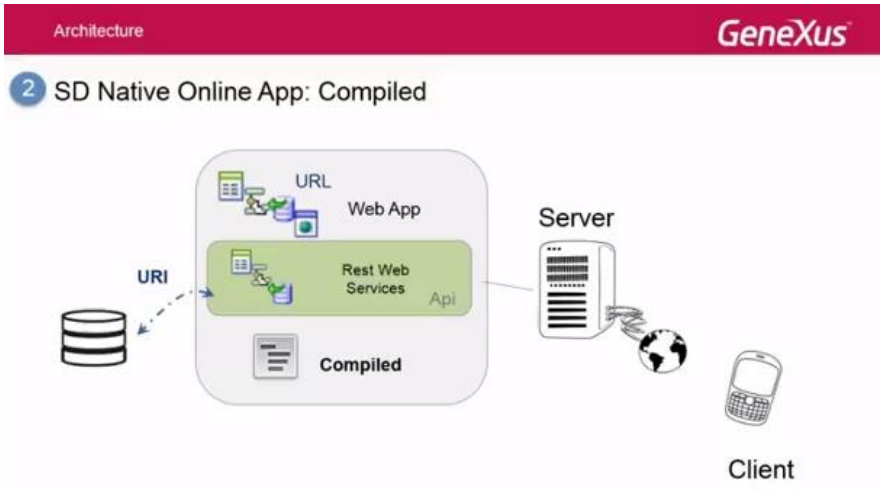
y devolverá al llamador el resultado de la operación



(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… como es el caso que estamos viendo.

Esto en lo que hace al KBN.

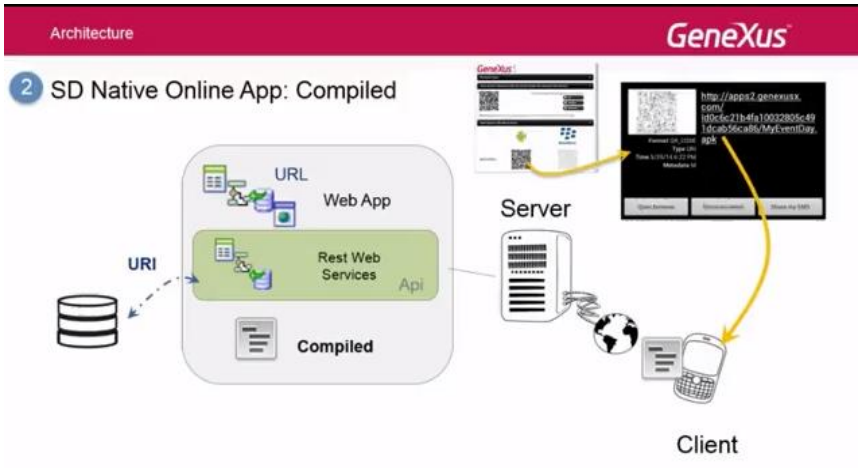
Ahora veamos la segunda opción, que de hecho será la que deberemos usar al final del ciclo necesariamente para poner en producción.



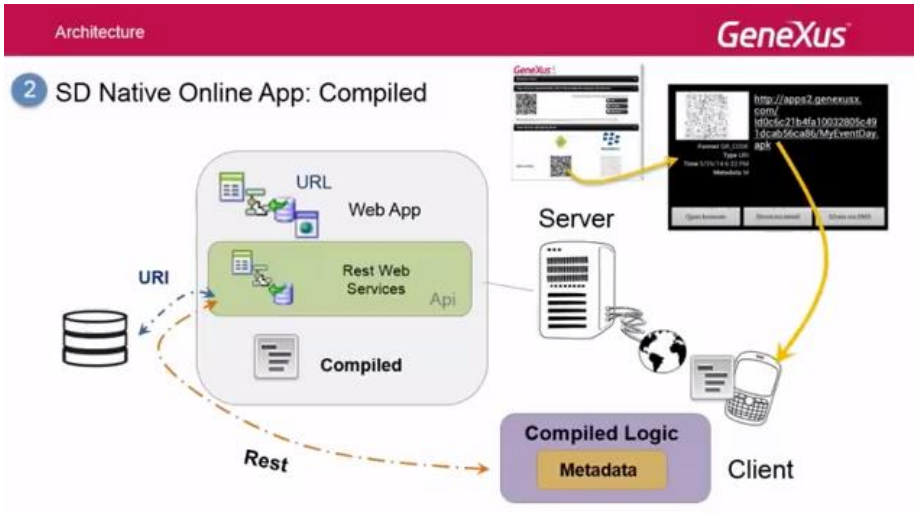
Compilar la aplicación.

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 necesitará del intérprete KBN, dado que es como un KBN customizado (que ya tiene configurada la URL de la aplicación), y encapsulará toda la metadata y las imágenes.

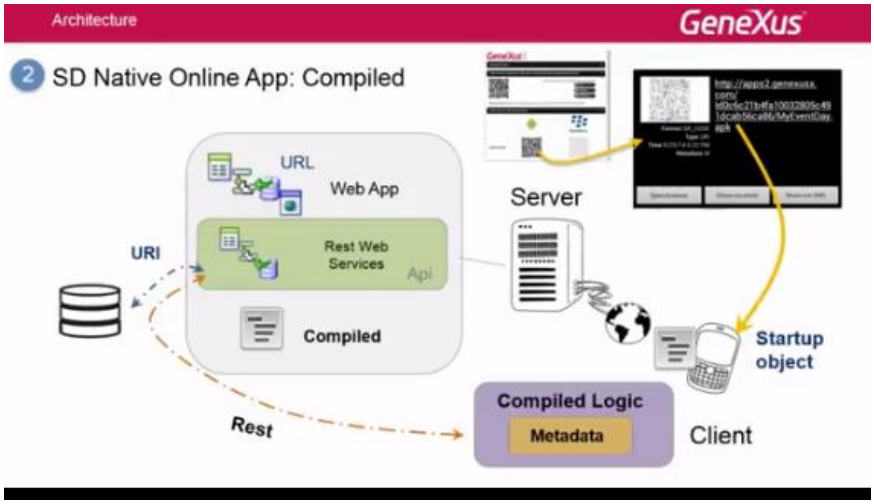


Por tanto, sólo deberá accederse al servidor para ejecutar los Rest Web Services que devolverán los datos actuales, procesados.



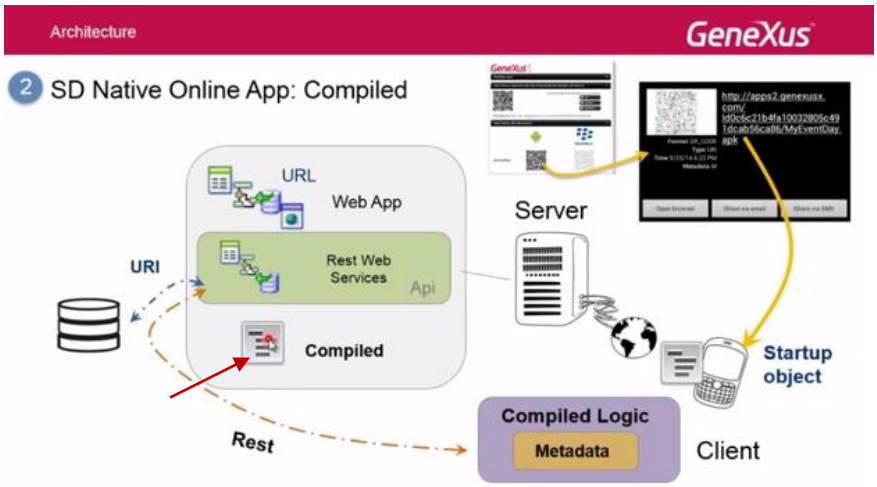
Para compilar la aplicación tenemos 2 posibilidades:

La 1era es indicar en las propiedades del Environment, **el startup object:**

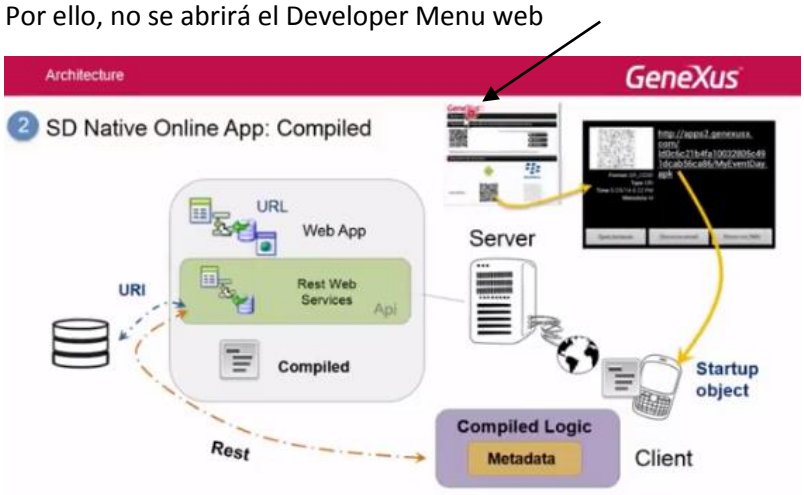


Por ejemplo en nuestro caso el dashboard MyEventDay.

Al hacer esto, tras el F5, no se generará la aplicación web, sino únicamente el compilado, apk, que se subirá al servidor.

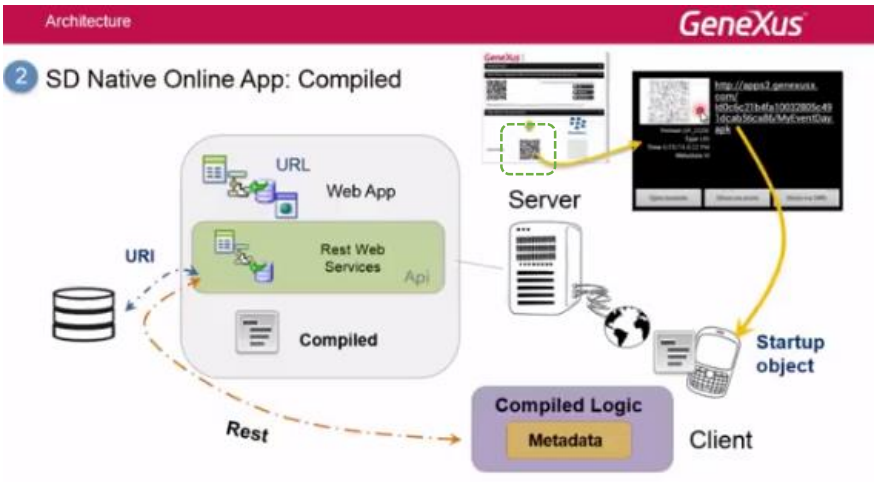


Por ello, no se abrirá el Developer Menu web



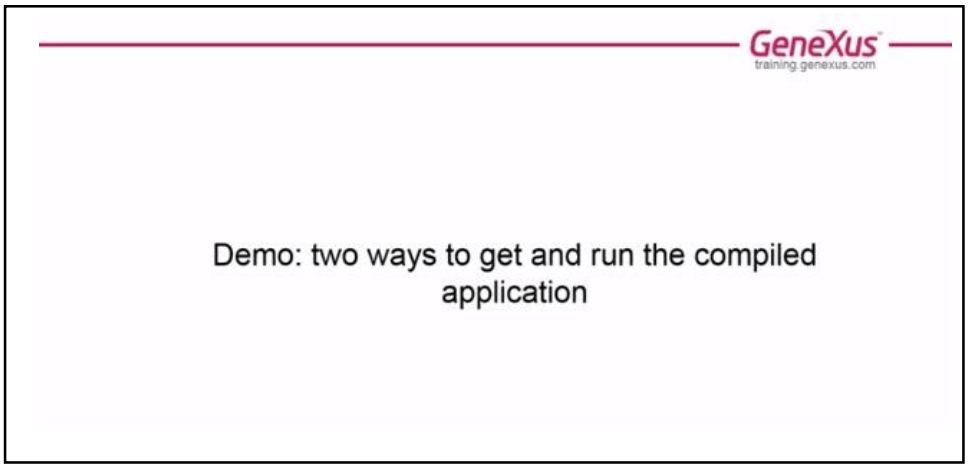
por lo que tendremos que pedirle explícitamente a GeneXus que nos lo abra a través de una opción que veremos en la demo que haremos en instantes.

Al hacerlo, si estamos prototipando en la nube, desde nuestro dispositivo, con el lector de QR codes que tendremos instalado



Podremos leer al QR Code que encapsula la URL donde se encuentra el compilado en el servidor de manera tal de poder descargarlo e instalarlo en nuestro dispositivo.

Veamos en GeneXus esta primera forma de obtener el compilado…y luego veremos allí mismo la segunda forma.



Video filmado con GeneXus X Evolution 3