Anexo: Comunicación entre aplicaciones y navegador Web en *Apps* de plataformas móviles

# El camino de ida: Desde el navegador Web hacia la *App*

En la mayoría de los sistemas operativos móviles, se permiten un mecanismo relativamente normalizado para la transferencia de control entre un navegador Web y una *App* en un sistema operativo móvil. Este modelo consiste en declarar el soporte de un esquema URL particular por parte de la *App* y la apertura de un enlace Web con este protocolo desde una página en el navegador Web.

Un ejemplo de este mecanismo en Apple iOS podría ser el soporte del protocolo *tel* en forma de URN con el formato [tel: 1-408-555-5555](tel:555-1234-4321), donde 1-408-555-5555 es un número de teléfono. Así una llamada desde una página Web a esta URN con una sentencia HTML como la siguiente:

<a href="tel:1-408-555-5555">1-408-555-5555</a>

Provoca que se active el teléfono (en un iPhone) y realice una llamada a ese número, ya que la aplicación nativa de teléfono de iOS tiene registrado ese esquema de protocolo.

Siguiendo con el ejemplo de iOS, el propio sistema operativo contiene ya una serie de aplicaciones que registran los protocolos más comunes, que para iPhone serían:

<http://developer.apple.com/library/ios/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html>

Si bien una llamada de tipo URN puede transferir el control desde el navegador Web hacia una *App*, no permite una forma avanzada de transferencia de información adicional, por lo que esta debe hacerse en la propia URN; así, en el ejemplo anterior vemos que en la URN se informa a la aplicación del número de teléfono al que queremos llamar. Otro ejemplo de llamadas tipo URN en las que se transfiere información adicional podría ser, continuando con Apple iOS, el envío de SMS, donde la llamada mediante el enlace:

<a href="http://maps.google.com/maps?q=cupertino">Cupertino</a>

Informa a la aplicación nativa que deseamos buscar la localización “cupertino” dando valor al parámetro “q” en la propia URL. En este ejemplo, aunque el protocolo es HTTP estándar, el sistema operativo discierne que siendo una llamada a maps.google.com debe ser atendida por la aplicación de mapas en vez de por el propio navegador.

La principal limitación en esta forma de transferencia de información es la cantidad de datos que podemos transferir. La información debe ser siempre textual y en caracteres soportados en URN (*URL Encoding*), lo cual obliga a que si queremos transferir un binario este deba ser antes convertido a Base64 (en una variante compatible con codificación *URL*). Si tenemos en cuenta que las cadenas de texto de las URL tienen una longitud limitada (que depende del sistema operativo y del navegador Web) y que un binario aumenta de tamaño al ser convertido a Base64 encontramos que hay una limitación práctica de unos pocos cientos de kilobytes.

Una estimación del tamaño máximo de caracteres que pueden ser transmitidos en una URL por Navegador Web es la siguiente:

* Internet Explorer: 2,083 caracteres
* Firefox: 65, 536 caracteres
* Safari: 80.000 caracteres
* Opera: 190.000 caracteres

Además, conviene tener siempre en cuenta que algunos sistemas operativos limitan esta transferencia de control entre navegadores Web y *Apps* por motivos de seguridad (supone un innegable riesgo), como ocurre por ejemplo en Windows Phone 7.5. En palabras de Isabel Gómez (Responsable de Aplicaciones y Juegos en Windows Phone, División de Desarrollo y Plataforma. Microsoft Ibérica):

“*Ese modelo no es viable en Windows Phone 7.5. Internet Explorer se ejecuta con privilegios bajos y por tanto no puede lanzar ninguna aplicación o ejecutable precisamente para evitar la entrada de virus o malware. La única opción es bien desarrollar todo en una App nativa o bien desarrollar todo en una aplicación web. No hay comunicación entre el navegador y las aplicaciones por diseño de la plataforma para evitar la introducción de virus/malware como ha ocurrido en el caso de Android.*”

# El camino de vuelta: Desde la *App* hacia el navegador Web

Una vez la *App* tiene el control del flujo de trabajo y ha realizado las operaciones apropiadas (por ejemplo, la firma electrónica, que no puede realizarse desde una página Web), sería deseable que el control fuese devuelto al navegador Web en el mismo punto donde lo cedió, de modo que el usuario pueda seguir con el flujo de su aplicación Web sin saltos ni necesidad de acciones adicionales.

El problema de esta vuelta hacia el navegador Web es que debe hacerse de la misma manera que en el sentido opuesto, mediante una llamada de apertura de una URN/URL con un protocolo concreto, que en este caso será el protocolo Web: HTTP o HTTPS, con una llamada de este estilo (ejemplo en Apple iOS):

NSURL \*url = [NSURL URLWithString:@"http://www.afirma.gob.es/back?param=sign1;sessionid=001"];

[[UIApplication sharedApplication] openURL:url];

Este mecanismo presenta ciertos problemas:

* ¿Qué pasa cuando hay más de un navegador Web? ¿Quién va a atender la petición? ¿Puedo garantizar que sea el mismo navegador que en un principio inició la transferencia de control?
  + El protocolo HTTP/HTTPS se atiende siempre por el navegador Web por defecto, con lo que será siempre este el que atienda la petición. Esto hace que si el usuario usó para acceder a la Web un navegador distinto al por defecto, el retorno abrirá un navegador distinto, causando una operativa confusa y molesta.
    - La imposibilidad de saber de antemano que aplicación va a atender un determinado protocolo queda claramente reflejada en el manual para programadores de Apple iOS:
      * ***Note:****If more than one third-party app registers to handle the same URL scheme, there is currently no process for determining which app will be given that scheme.*
      * <http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/AdvancedAppTricks/AdvancedAppTricks.html>
  + Adicionalmente algunos sistemas operativos no permiten variar el navegador Web por defecto, como ocurre en Apple iOS que será siempre Safari quien responda a apertura de URL con protocolo HTTO o HTTPS.
* En el caso de que el usuario únicamente use el navegador por defecto (aspecto que ni mucho menos puede darse por supuesto) ¿Podemos estar seguros de continuar con la sesión que teníamos iniciada? De nuevo, la respuesta es negativa. En muchos sistemas operativos la petición se atenderá en una nueva pestaña del navegador (por ejemplo, Apple iOS 5 y superiores), con la mala experiencia de usuario que eso conlleva.
  + Si a este aspecto sumamos el hecho de que una *App* no tiene acceso a las Cookies del navegador, obliga a una gestión de sesiones mediante parámetros en URL, introduciendo incluso más limitaciones molestas.
    - Según documentación de Apple iOS:
      * ***iOS Note:****Cookies are not shared by applications in iOS.*
      * <http://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/URLLoadingSystem/Concepts/URLOverview.html>

Adicionalmente, seguimos contando con las limitaciones que encontrábamos en el sentido inverso:

* La cantidad de información que se puede transferir es muy limitada.
* Algunos sistemas operativos (como Microsoft Windows Phone 7.5) no permiten esta operativa por motivos de seguridad.

# Conclusiones

Las limitaciones actuales que se imponen para la comunicación entre *Apps* en dispositivos móviles (siendo el navegador Web una *App* más) no son casuales, son restricciones impuestas por motivos de seguridad. Aun estando la operativa limitada, se ha demostrado ser fuente de ataques maliciosos por parte de *hackers* y piratas, especialmente notables en Google Android.

Es por lo tanto fácil aventurar que estas limitaciones no pueden sino endurecerse en las versiones futuras de los sistemas operativos móviles:

* La seguridad prima sobre la funcionalidad en este caso, y los fabricantes impondrán las medidas que consideren necesarias para garantizar una mínima seguridad. Esto puede hacer que operativas que actualmente funcionan dejen de hacerlo en nuevas versiones.
* Además, el mercado de los navegadores Web en plataformas móviles están en plena expansión (Firefox móvil, Google Chrome, Opera, Safari, Pocket Internet Explorer, etc.). Limitar al usuario la elección del navegador sin duda es una decisión que puede causar rechazo en los usuarios, además de problemas por desconocimiento.

La solución más adecuada pasa por reducir al máximo la interacción entre navegador Web y la *App* (@firma móvil en este caso), usando una llamada “sin retorno” que no corra el riesgo de intentar devolver el control al navegador. Las pruebas mediante de prototipos que han realizado, dentro del proyecto @firma, entidades como la Junta de Andalucía y Atos han constatado que los problemas en el retorno de control de una *App* al navegador Web son reales y graves.

Por supuesto, hay alternativas de operación. La más inmediata es la anteriormente comentada limitación al usuario para que use un único navegador Web (que además puede ser necesario que sea el “por defecto”) para ejecutar las aplicaciones. Esta aproximación, si bien puede ser útil en entornos muy controlados (por ejemplo, es la utilizada en las aplicaciones Google Android de las tabletas Samsung Galaxy del Senado), es demasiado intrusiva para los usuarios ciudadanos como finales (¿Por qué tendría una administración pública decirle al usuario de un Apple iPad que use Safari en vez de Opera? <http://itunes.apple.com/es/app/opera-mini-web-browser/id363729560>).

# Opciones de futuro

La transferencia de control mediante registro de protocolos y apertura de URN/URL no es la única forma de ejecutar código nativo desde una página Web. Si el dispositivo contase con un servidor Web propio con capacidad de ejecutar código invocado mediante URL una aplicación JavaScript dentro de una página Web en un navegador podría realizar esta comunicación mediante JSON y servicios web tipo REST con llamadas a *localhost* e implementando medios de autenticación para garantizar la seguridad.

¿Es este modelo viable? Tecnológicamente hay grandes variaciones entre sistemas operativos, mientras que Android permite el desarrollo e instalación de servicios en segundo plano (el servidor Web propio en local sería uno) Apple iOS y otros sistemas operativos (como Windows Phone 7.5) lo limitan por motivos de seguridad, debiendo hacer uso del sistema integrado de notificaciones, lo cual impide implantar este modelo con eficiencia (Apple iOS: <http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Introduction/Introduction.html>).

Otro problema adicional es que un servicio en segundo plano que quede escuchando a la espera de peticiones (de firma en este caso concreto), queda también consumiendo procesador, memoria y lo que es incluso más importante, batería).

Es por lo tanto que esta supone una opción a explorar, pero no se considera como opción prioritaria para una implementación de @firma móvil a corto plazo.

**Autor:**

Tomás García-Merás

[tomas.garciameras@atos.net](mailto:tomas.garciameras@atos.net)