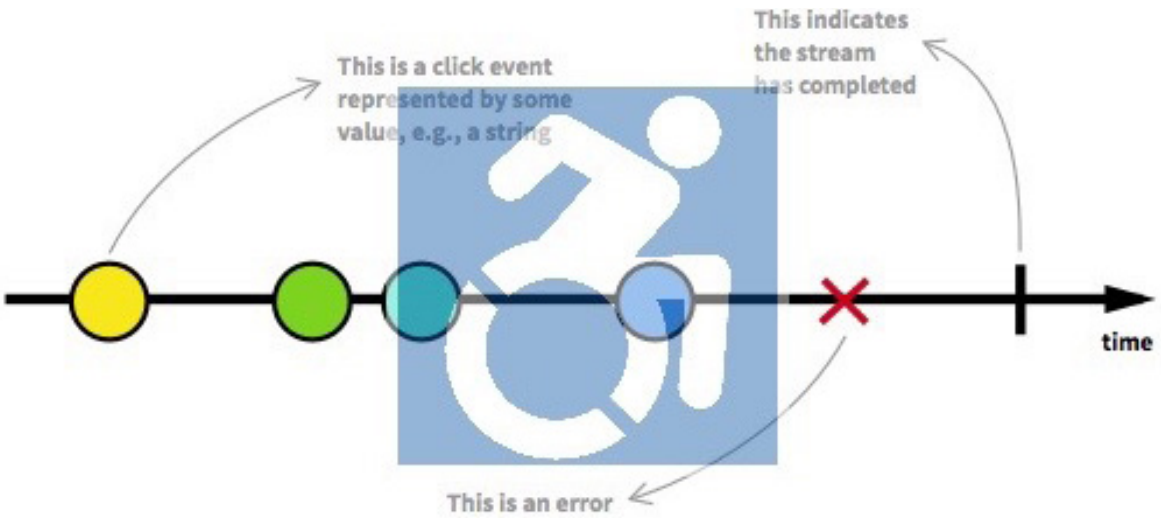
Accesibilidad y los problemas con la programación reactiva

Jonathan Chacón
Socio de Asepau

Dentro de los distintos paradigmas existentes en el mundo de la programación el más beneficiado por la aparición de interfaces de usuario gráficas ha sido el paradigma de la programación orientada a objetos, o POO, en la cual un gran problema se divide en muchos problemas más pequeños y se articulan como objetos. Cada objeto es responsable de recibir una información, gestionarla y proporcionar un resultado.

Con la programación orientada a objetos cada elemento gráfico de la interfaz de usuario se trataba como un objeto. Un botón, un cuadro de texto, una etiqueta de texto o una barra de progreso eran distintas clases de objetos y se manipulaban siguiendo una programación imperativa que seguía las reglas de la programación orientada a objetos.

Otros objetos nos permiten, por ejemplo, enviar un correo electrónico, consultar el estado de conectividad del dispositivo o simplemente modelar una entidad de nuestro modelo de datos. Un paradigma muy potente con sus ventajas e inconvenientes.



Diferencias entre paradigma de programación y lenguaje de programación

No debemos confundir un paradigma de programación con un lenguaje de programación.

Un lenguaje de programación es el conjunto de instrucciones, de palabras reservadas y su sintaxis, que proporciona al programador la forma de interactuar con un ordenador para indicarle las acciones que debe ejecutar.

Un paradigma de programación es un modelo básico de diseño y desarrollo de aplicaciones, que permite producir programas con un conjunto de normas específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc.

El mundo evoluciona

Aunque el paradigma de la programación orientada a objetos ha reinado por más de 30 años en la actualidad convive con otros paradigmas como la *programación declarativa*, la *programación orientada a aspectos* o la *programación reactiva*.

Aunque un lenguaje de programación se oriente a un paradigma determinado de programación en la actualidad los lenguajes más modernos pueden hacerse compatible con otros paradigmas de programación con ampliaciones al lenguaje utilizando librerías o APIs.

Programación reactiva

La programación reactiva es uno de los paradigmas con más éxito en la actualidad. Muchos desarrolladores de software y empresas apuestan por este modelo de diseñar y programar aplicaciones para dispositivos móviles.

La programación reactiva es un paradigma que se enfoca en trabajar con flujos asíncronos de información. Estos flujos de información pueden contener eventos (el usuario toca algo, la aplicación se cierra, se termina de descargar algo de la Nube, etc), bloques de datos o mezcla de ambos. Estos flujos de información pueden ser finitos o infinitos.

Su concepción y evolución ha ido ligada a la publicación del Reactive Manifesto, que establecía las bases de los sistemas reactivos. Estos sistemas deben cumplir los siguientes requisitos:

- **Responsivos:** deben garantizar el cumplimiento de unos tiempos de respuesta de la aplicación para asegurar la calidad del servicio.

- **Resilientes:** deben mantenerse responsivos incluso ante situaciones de error.

- **Elásticos:** deben mantenerse responsivos incluso ante aumentos en la carga de trabajo.

- **Orientados a mensajes:** deben minimizar el acoplamiento entre componentes al establecer interacciones basadas en el intercambio de mensajes de manera asíncrona.

Entre las ventajas de la programación reactiva frente a otros paradigmas destacan la posibilidad de controlar la aplicación de forma asíncrona, a diferencia de la POO donde todo es síncrono. El control asíncrono es ideal en un mundo donde comunicarse con la nube, esperar respuestas y datos de orígenes remotos o la posibilidad de que el usuario cambie el flujo de uso de la aplicación bien porque el usuario recibe una llamada telefónica o una notificación de su aplicación de mensajería instantánea. En un modelo síncrono el programador debe ir introduciendo cláusulas de pausa o salida en todos los procesos por si se produce la interrupción. En un modelo asíncrono es el propio sistema quien controla salir del estado y retomarlo cuando el usuario vuelva a la aplicación.

Es claro para cualquier programador del hecho de que la programación reactiva ha llegado para quedarse bastante tiempo debido a los movimientos de grandes fabricantes como Apple, Microsoft o Google que están creando y publicando librerías y APIs orientadas a la programación reactiva.

Qué es una API

Una API (Application Programming Interface) es un conjunto de definiciones, funciones y recursos de programación (código y protocolos de comunicación) que permite a una aplicación utilizar o integrar recursos y funcionalidades de otras aplicaciones o módulos del sistema operativo.

Existen APIs para controlar la interfaz gráfica de usuario, comunicarse con otros dispositivos, acceder a los recursos de la máquina, etc.

La diferencia entre una librería y una API es que la API es sólo la declaración de funciones y propiedades para alcanzar unos objetivos. La librería además de integrar la declaración del conjunto de funciones incluye la codificación para alcanzar los diversos objetivos que permiten las funciones agrupadas dentro de la librería. Por buscar un simil, con la API tenemos la carta del menú y con la librería tenemos además las herramientas e ingredientes de la cocina del restaurante.

A veces a un programador le es suficiente con usar las APIs ya que su aplicación será más ligera ya que todo el código relacionado con las APIs que use no irán empaquetados con su aplicación, es el sistema operativo quien debe proporcionar el código a ejecutar por las funciones llamadas desde la aplicación utilizando estas APIs.

En cambio, a veces el programador necesita modificar el comportamiento de algunas funciones de una API, debe incluir dentro del paquete de su aplicación el código necesario para provocar el cambio de comportamiento de esas funciones modificadas provocando que su aplicación tenga un mayor tamaño a la hora de ser instalada en un dispositivo.

Las API de accesibilidad y la POO

La programación orientada a objetos (POO) ha reinado durante más de 30 años en el mundo de la programación. Tanto que casi todas las API de accesibilidad que existen para distintos sistemas operativos se han diseñado siguiendo este paradigma.

Un objeto de la interfaz gráfica de usuario posee diversas propiedades como su color, su posición, su contenedor, sus dimensiones, entre estas propiedades hay algunas relacionadas con la accesibilidad: un posible texto alternativo, un área de activación, un rol, un estado, un identificador para productos de apoyo, etc.

Dependiendo de lo madura y completa que sea la API de accesibilidad esta proporcionará más propiedades y funciones que otras. Por ejemplo, las API de accesibilidad de iOS y Windows son más completas que las de Android y MacOS aunque los fabricantes siguen ampliando estas APIs para proporcionar interfaces cada vez más accesibles.

Un problema de las API de accesibilidad de muchos sistemas operativos es la inclusión por defecto, como sucede en iOS y MacOS, o la inclusión por esfuerzo, como sucede en Android. Esto implica que si un desarrollador de software usa un botón de la librería estándar de objetos gráficos de su sistema operativo en iOS

y Windows el botón será accesible por defecto pero en Android el desarrollador deberá esforzarse por incorporar accesibilidad en su botón.

Mientras más se aleje del botón estándar la interfaz creada por el desarrollador más deberá esforzarse por mantener la accesibilidad. Esto es común para todos los sistemas operativos. Por esta razón es muy recomendable utilizar objetos de interfaz gráfica pertenecientes al estándar de cada plataforma.

Definiendo la accesibilidad con POO

A la hora de definir una interfaz accesible utilizando POO era necesario definir cada propiedad de un objeto, incluyendo sus atributos de accesibilidad.

Ejemplo de un botón con mucha información

Para este artículo tendremos como ejemplo una app que muestra la información climática de varias ciudades. Toda esta información se obtiene de un servidor externo en la Nube.

La app se abre y mientras muestra la interfaz de bienvenida llama a la Nube para que le envíe la información de cada ciudad.

Imaginemos que en nuestra aplicación por cada ciudad tenemos un botón que nos permite consultar el tiempo en una localización determinada.

Como etiqueta para nuestro botón tenemos los siguientes elementos:

· Una etiqueta de texto con el nombre de la ciudad
· Un icono a la izquierda con una imagen del estado del clima (sol, nubes, lluvia, etc)
· Una etiqueta de texto a la derecha con la temperatura en grados celsius

Nuestro botón, al tratarse de una app nativa, es recomendable que utilice la propiedad para definir un texto accesible. Este texto accesible deberá describir toda la información que muestra el botón, por ejemplo: **Soleado, Málaga, 24**. Incluso podemos componer la información para la etiqueta de accesibilidad en algo más comprensible para un humano que usa lector de pantallas dejando nuestra etiqueta con el texto: **Málaga, soleado y 24 grados.**

La etiqueta de accesibilidad es una propiedad del objeto Botón que puede definirse en cualquier momento como una cadena de texto que representa al botón dentro de la API de accesibilidad.

En android estamos hablando de la propiedad *ContentDescription* y en iOS se trata de la propiedad *AccessibilityLabel*. Estas propiedades serán verbalizadas por el lector de pantallas cuando se focalice el botón.

Al recibir la información de la ciudad desde la Nube el programador debe actualizar de forma manual el icono, las dos etiquetas de nombre de la ciudad y la temperatura. Además, deberá actualizar el valor de la etiqueta de accesibilidad.

El programador está acostumbrado a trabajar de esta forma actualizando las diversas propiedades de un objeto para reflejar el estado de la aplicación en todo momento.

El proceso de forma simplificada sería algo como:

· Llamada a la nube para solicitar información de las ciudades
· Recoger la información de cada ciudad
· Actualizar manualmente cada botón para mostrar la información correcta

Si el programador estuviese desarrollando su app utilizando un paradigma o unas librerías de programación reactiva el proceso sería un poco distinto:

· Definir un modelo de información para el flujo de información
· Llamada a la nube para solicitar información de las ciudades
· Conectar cada botón con el flujo de información de la llamada realizada
· Esperar a que se reciba la respuesta y la programación reactiva haga el resto

Con la programación reactiva es más sencillo para el programador a la hora de mantener actualizada la interfaz de usuario y los diferentes estados de los controles de interfaz. Pero ciertas propiedades de los objetos de interfaz no son actualizadas de forma automática por las librerías de programación reactiva por lo que, si el programador no lo sabe, estas propiedades no serán actualizadas nunca.

Imaginemos que nuestra aplicación de ejemplo, al arrancar, carga los últimos datos que obtuvo el día anterior en la que Málaga tenía un clima soleado de 24 grados Celsius pero hoy en Málaga el día está nublado y con una temperatura de 12 grados celsius.

En una versión con programación reactiva la interfaz de usuario, visualmente, se actualizará sin problemas pero la parte de accesibilidad no se actualizará de forma automática a menos que el programador lo haya tenido en cuenta.

Y el uso de controles de interfaz de usuario de la librería estándar no es garantía debido a que las librerías de programación reactiva existentes en la actualidad para aplicaciones de dispositivos móviles no contemplan la actualización de atributos y estados de accesibilidad.

Un programador que desconozca las peculiaridades y las necesidades de accesibilidad encontrará que aunque utilice botones de la librería estándar sus controles dejan de ser accesibles creando nuevas barreras de accesibilidad.

Además estas nuevas barreras son más difíciles de detectar en el caso de sistemas operativos con accesibilidad por defecto, como es el caso de iOS, ya que los usuarios de lectores de pantalla encontrarán que todos los controles tienen etiquetas accesibles pero no sabrán que los controles se han actualizado con nuevos valores.

Es conocido el desconocimiento mayoritario de los desarrolladores de software sobre las necesidades y características de accesibilidad que ofrecen las diversas plataformas y sistemas operativo. Los fabricantes han ido aportando soluciones automatizando el proceso de incorporar accesibilidad por defecto a los controles de interfaz de usuario de las librerías del estándar de cada plataforma pero ahora nos enfrentamos al problema de que la interfaz dejará de ser accesible debido a que la accesibilidad se definirá al principio de mostrarse la interfaz pero no se actualizará a posteriori. Este hecho puede traer en muy poco tiempo un retroceso en la accesibilidad de las aplicaciones para dispositivos móviles.

Aunque Apple, con su librería *Combine* y Microsoft con la actualización de librerías de controles de interfaz de usuario están buscando una solución para la actualización asíncrona de los atributos y valores relacionados con la accesibilidad de las interfaces de usuario hoy por hoy vuelve a caer toda la responsabilidad de la accesibilidad en los programadores.

Esperemos que en poco tiempo la accesibilidad también sea asíncrona, reactiva y más conocida por la comunidad de programadores para alcanzar cuotas cada vez mayores de accesibilidad e inclusión en el mundo del software.