Creating and Using Proximity Events on Mobile Websites

P. M. Vera and R. A. Rodríguez

***Abstract*— Nowadays mobile devices includes a large variety of hardware components and sensors which are used by several applications for obtaining context information or creating new ways of interaction. At the beginning hardware access was limited for installed applications only (native applications) but day after day new web standards arises allowing accessing hardware component and sensor. Geolocation, for example, can be used from web sites in the same way and with equivalent results than in a native application. This article analyses the proximity sensor API (Application Programming Interface) and proposes it use for the detection of different events that can be used in a web environment. For example for improving navigation by going back to the home page or opening a menu by simply moving a finger over the sensor.**

***Keywords*— Mobile Devices, Cell Phones, Sensors, Proximity Sensor, HTML 5, API**

I.INTRODUCCION

TODOS los dispositivos móviles tienen ciertas limitaciones al navegar por la web y en especial los teléfonos celulares debido a sus pantallas reducidas. Esto hace que sea necesario diseñar sitios web especialmente adaptados.

El W3C (World Wide Web Consortium) brinda una serie de buenas prácticas dirigidas al diseño de los sitios web móviles [1], [2] e incluso herramientas de validación automática que permiten evidenciar problemas en los sitios web construidos [3], [4], [5]. Siguiendo las recomendaciones de estas guías es posible desarrollar sitios web que sean usables desde dispositivos móviles. No obstante es posible, a pesar de haber considerado las recomendaciones del W3C no lograr construir sitios con un contenido adecuado respetando la usabilidad. Por ello existen diversos autores que complementan dichas guías con cuestiones referidas a la usabilidad, entre ellos Nielsen & Budiu [6], Krug [7]. En general el problema principal es que los sitios web son extensos y es necesario realizar demasiado scroll para volver a la parte superior o inferior donde la mayoría de los sitios incorporan las opciones de navegación. Hay que considerar que cualquier objeto que se agregue en la pantalla ocupará valioso espacio por lo que dejar las opciones fijas en pantalla o dejar un menú lateral no son opciones viables.

En cuanto a la barra de navegación, el W3C recomienda situarla en la parte superior de la pantalla [1], dicha barra tiene que ser reducida además para minimizar el espacio. Pero más allá de las recomendaciones del W3C, la mayoría de los diseños incorporan menues de navegación más completos, lo que por otro lado evita tener que volver a cargar páginas anteriores para alcanzar otro contenido. Estos menús son implementados de diferentes formas:

- Mostrando directamente las opciones en la página: lo que ocupa mucho lugar y reduce ampliamente la usabilidad.
- Usando un menú desplegable mediante el uso de un control estilo dropdown, lo que si bien ahorra lugar, no es muy recomendable a la hora de encontrar una opción. Además tampoco permite mostrar jerarquías.
- Usando un menú colapsable que se representa en pantalla mediante un link o un ícono que muestra el menú jerárquico, no ocupando lugar ya que solo se visualiza a pedido del usuario. Siendo esta opción la más recomendable.

En general todas las opciones de navegación se encuentran en la parte superior (cumpliendo con lo sugerido por la W3C) lo que obliga al usuario a tener que hacer scroll para acceder a la misma. Este trabajo propone mejorar esta cuestión de usabilidad haciendo uso del sensor de proximidad disponible actualmente en todos los teléfonos con pantalla táctil y en general en todos los smartphones. El cual puede ser utilizado desde una página web gracias a HTML 5.

II. SENSORES

Actualmente los smartphone tienen una gran cantidad de sensores y componentes de hardware [8]. Los cuales se muestran en la Fig. 1. Existen aplicaciones que pueden instalarse en el dispositivo y detectan qué sensores están incorporados en un equipo.

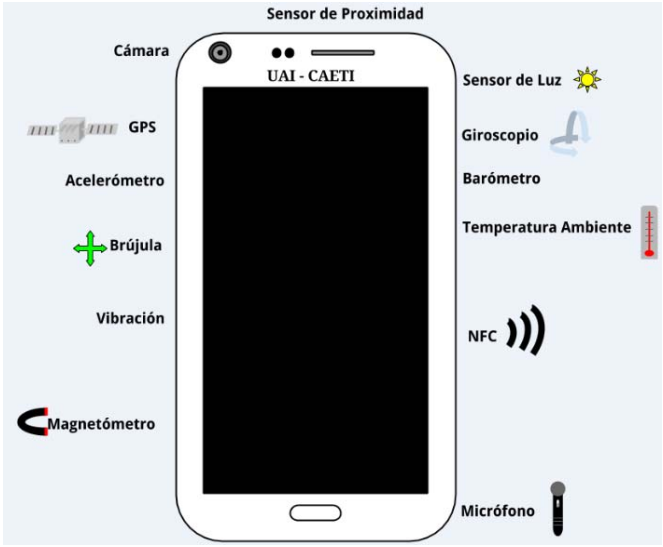

Figura 1. Sensores y otros componentes de Hardware que pueden estar incorporados en un Smartphone.

P. M. Vera, Universidad Abierta Interamericana, Ciudad Autónoma de Buenos Aires, Argentina, PabloMartin.Vera@UAI.edu.ar

R. A. Rodríguez, Universidad Abierta Interamericana, Ciudad Autónoma de Buenos Aires, Argentina, RocioAndrea.Rodriguez@UAI.edu.ar

Corresponding author: Pablo Martín Vera

III. SENSOR DE PROXIMIDAD

Permite detectar cuando un objeto está cerca del dispositivo, generalmente este sensor se utiliza para apagar la pantalla cuando en una llamada se acerca el equipo al oído o cuando se lo guarda en una funda. Para no interferir cuando se toma el equipo con las manos este sensor generalmente se ubica en la parte superior de la pantalla lo que es conveniente ya que es posible acceder a él fácilmente. Por ejemplo teniendo el teléfono con una sola mano es posible tapar el sensor de proximidad simplemente moviendo el dedo pulgar hacia arriba. Esta característica permite accionar rápidamente el sensor y poder capturar eventos sobre el mismo como si se tratara de un botón físico. Aprovechando estos eventos es posible incorporar a cualquier sitio web, ayudas a la navegación, para ser usadas como alternativas a las tradicionales sin ocupar espacio en la pantalla.

El sensor de proximidad solo mide la presencia o no de un objeto cercano, no mide la distancia. Por lo tanto como parte de este trabajo se ha desarrollado una función en javascript que permite capturar ciertos eventos basados en pasar el dedo sobre el sensor obstruyéndolo y liberándolo (es decir pasando sobre el sensor). En base a esto se podrán detectar los siguientes eventos: pase corto; pase largo; doble pase.

Para ello la función mide los tiempos desde que se detecta la presencia del objeto cercano, hasta que se deja de detectar debido a que se ha alejado. El sensor retorna un 0 cuando detecta el objeto y >0 cuando no lo detecta (ver Fig. 2).

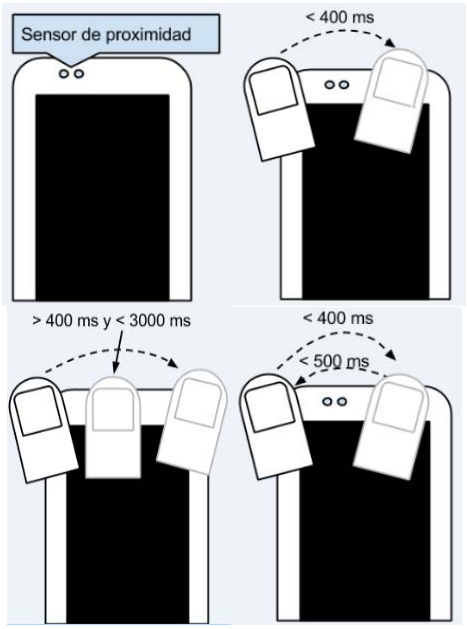

Figura 2. Sensor de Proximidad y los pases: Corto, Largo y Doble.

Luego de varias pruebas se han establecido los tiempos para la detección de cada evento:

- Para detectar un pase corto se ha determina un tiempo inferior a 400 milisegundos.
- El pase largo mayor a 400 y menor a 3000, para evitar activar el evento si por ejemplo se guarda el teléfono en la funda.
- El doble pase se detecta si luego de un pase corto se realiza un segundo pase corto dentro de los próximos 500 milisegundos.

Los valores considerados en principio son arbitrarios simplemente tomando en cuenta la experiencia de uso en las pruebas durante el desarrollo de la función javascript. Más adelante en este artículo se expondrán las pruebas realizadas, los ajustes de los valores y los problemas detectados.

IV. PROPUESTA DE USO

Teniendo disponible este sensor y la función para detectar los eventos se proponen entonces implementar las siguientes ayudas a la navegación:

- Utilizar la doble pasada: por ejemplo para mostrar en un menú desplegable las opciones de navegación, esto evitará tener que hacer scroll para moverse hasta donde se encuentre el menú o el ícono que abre el mismo. El menú desplegable podrá a su vez cerrarse mediante una pasada corta. Otra opción sino se desea utilizar en menú desplegable es que la doble pasada lleve automáticamente a la parte superior de la pantalla donde se encuentran las opciones.
- Utilizar la pasada larga: para volver al "home" del sitio web. Creando un acceso directo desde cualquier lugar del sitio para ir a la página principal.

Estos ejemplos de uso dejan en evidencia las alternativas adicionales que se le ofrecen a los usuarios finales, mediante el uso del sensor de proximidad.

A. Creación de una biblioteca de JavaScript

Se desarrolla una biblioteca javascript que podrá ser consumida desde los distintos sitios web para aprovechar los beneficios del uso del sensor de proximidad en la usabilidad de los mismos.

B. Uso de la Biblioteca Javascript

1. Referenciar la biblioteca de javascript dentro del head del documento html

```
<script src="UAIProximityEvents.js">

</script>
```

2. Iniciar el sensado mediante la función StartSensing. Esta función recibe tres parámetros, permitiendo seleccionar el tipo de evento a detectar y si se desea o no recibir feedback mediante vibración, los parámetros son los siguientes:

```
function StartSensing(blnSingle, blnDouble, blnLong, blnVibrate)
```

Donde cada uno de los parámetros deben ser valores booleanos habilitando o deshabilitando la característica deseada:

- blnSingle: detecta y dispara un evento ante un pase corto por el sensor de proximidad.
- blnDouble: detecta y dispara un evento ante un doble pase por el sensor de proximidad.
- blnLong: detecta y dispara un evento ante un pase largo sobre el sensor de proximidad.
- blnVibrate: habilita la vibración como feedback ante la detección de cada uno de

los eventos. La vibración será diferente según el evento detectado siendo una vibración corta, doble o larga; respectivamente para cada uno de los eventos.

3. Por cada uno de los eventos habilitados se debe crear en la página un listener, dentro de los cuales se deberá colocar el código correspondiente a la acción deseada para cada uno de los eventos.

```
document.addEventListener('ProxSingleEvent',
function (e) {

            }, false);

document.addEventListener('ProxDoubleEvent',
function (e) {

            }, false);

document.addEventListener('ProxLongEvent',
function (e) {

            }, false);
```

V. PRUEBAS REALIZADAS Y PROBLEMAS DETECTADOS

La API realizada fue probada mediante dos etapas, la primera consistió en evaluar la usabilidad de la misma, para lo cual se seleccionó un equipo puntual y se les solicitó a 5 usuarios que realicen una cierta cantidad de pases cortos, largos y dobles para ver si lo lograban y los tiempos de detección. "Los mejores resultados pueden obtenerse por medio de pruebas de no más de 5 usuarios y corriendo tantas pruebas distintas como pueda permitirse" [9]. Una vez configurados los tiempos en base a los resultados obtenidos, se les pidió a otro conjunto de usuarios que prueben en sus dispositivos (en equipos con acceso a internet y browser Firefox) realizar los pases cortos, largos y dobles con los ajustes realizados luego de la primera etapa de pruebas. A continuación se presenta brevemente los detalles de las pruebas realizadas y los resultados obtenidos.

A. Etapa I – Independencia del Dispositivo.

Primeramente, se tomó un dispositivo particular (Samsung S3 Mini) y se les pidió a 5 usuarios que realizarán cierta cantidad de pases cortos, largos y dobles. En la literatura puede encontrarse fundamentación sobre la cantidad de usuarios indicados para pruebas en la web [9], [10].

Los problemas encontrados fueron que los pases cortos podían confundirse con largos, por ello es importante configurar las cotas de tiempo (milisegundos - ms). La Fig. 3 muestra los resultados en ms obtenidos por 5 usuarios que realizaron pases cortos. Pudiendo notarse que ninguno de los pases cortos supera los 200 ms y en su mayoría están comprendidos entre 20 y 100 ms.

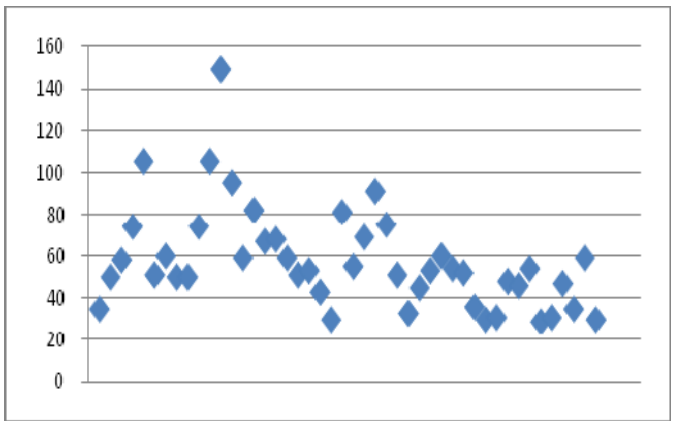

Figura 3. Tiempos registrados en pruebas de pases cortos.

En cuanto a los pases largos (Fig. 4), si se define con poco tiempo puede interpretarse como un pase corto por ejemplo aquellos que se encuentran hasta los 200 ms. Y por otra parte poner una cota superior alta hará que los usuarios esperen en demasía para ver el evento en la pantalla producto a ese pase largo.

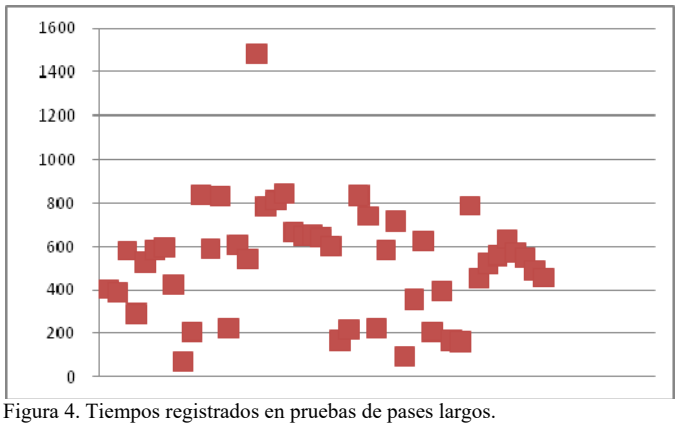

Figura 4. Tiempos registrados en pruebas de pases largos.

Producto de este análisis fue posible: Configurar la cota más alta de tiempo de espera para la detección de un pase largo (agregando vibración para que el usuario sepa que el evento fue detectado). Se ajustaron las cotas de los pases largos y cortos, para evitar errores de detección.

B. Etapa II – Pruebas con Distintos Dispositivos.

En una segunda etapa se realizaron pruebas en distintos equipos. Cabe destacar que la implementación de uso del sensor de proximidad desde la web en HTML 5, funciona únicamente para el browser Firefox [11]. Con lo cual excluye de las posibles pruebas a dispositivos que tengan sistemas operativos como IOS y WindowsPhone dado que aún no hay versiones disponibles para estos sistemas operativos del browser.

Producto de esta prueba pudo notarse que los sensores de proximidad tienen diferente grado de sensibilidad dependiente de la marca del equipo. No siempre el sensor es lo necesariamente sensible como para detectar un pase corto realizado a gran velocidad, lo mismo ocurre con el pase doble. Las pruebas se realizaron con equipos de marcas: HTC (en un

dispositivo: ONE), LG (en 2 equipos: L4, Nexus 4), Sansumg (en 4 equipos: Galaxy, S3 Mini, S4 y S4 Mini). No funcionó en los equipos LG y si funcionó correctamente en el resto de los equipos de prueba. El no funcionamiento puede deberse al hardware del sensor así como su configuración para la detección de los eventos. El problema es que el acceso a los menues de configuración de este tipo de hardware suelen estar ocultos requiriendo que el usuario ingrese un código específico según el modelo de su celular para poder acceder a dicho menú (ver Fig. 5), además en algunos casos estos menues sólo tienen opciones de testing y no de configuración. Se descarta la opción de solicitar a un usuario que deba hacer este tipo de configuración para poder ajustar el sensor de proximidad.

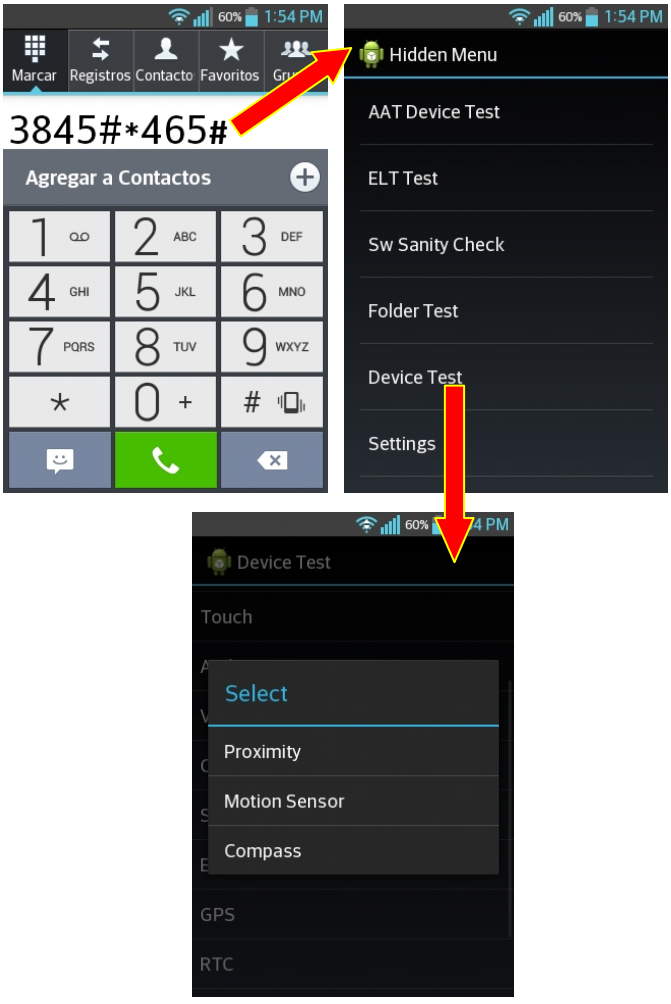

Figura 5. Acceso al menú oculto de testing/configuración del sensor de proximidad desde un LG.

VI. AJUSTES EFECTUADOS

El esquema de uso anterior no era adecuado para todos los tipos de sensores utilizados en los dispositivos debido a que algunos de ellos no disponen de la velocidad suficiente de sensado como para poder capturar dichos eventos, aun aumentando los tiempos, los sensores no son los suficientemente rápidos a los cambios. Por eso se cambia la estrategia de utilización, detectando ahora el tiempo en el cual el sensor permanece bloqueado, configurando un tiempo base y detectando las unidades de tiempo que se mantiene obstruido el sensor. De esta forma se desarrolla una nueva API la cual dispara un evento al dejar de obstruir el sensor indicando en una variable la cantidad de unidades de tiempo detectadas. Por cada unidad de tiempo se da un feeback al usuario mediante la función de vibración del teléfono. También puede configurarse la cantidad máxima de unidades de tiempo que se desea detectar haciendo que al llegar a ese número el evento se dispare independientemente de si el usuario deja de obstruir el sensor de proximidad o no.

El usuario tendrá total conocimiento de en qué momento puede mover su dedo debido a que las vibraciones le indicarán que evento estará por ejecutarse, por ejemplo: 1 ir al final de la página, 2 ir arriba de la página, 3 volver a la página principal. Es importante que el usuario tenga tiempo de mover su dedo y dejar de obstruir el sensor para que no se contabilicen más unidades y de este modo se interprete otra acción.

A. Nueva API

Para adaptarse a las distintas velocidades de los sensores y poder realizar una API que funcionara en cualquier dispositivo fue necesario realizar un cambio de paradigma en la forma de detección planteada. Ya no es posible detectar con velocidad un pase corto o un doble pase sino que lo que se medirá es el tiempo que el usuario obstruye el sensor de proximidad. Pudiendo asignar una unidad de tiempo estándar, es posible detectar cuántas unidades de tiempo el usuario obstruye el sensor permitiendo así asignar distintas acciones según dicho tiempo.

Para utilizar la api se deben realizar los siguiente pasos:

1. Referenciar el archivo con las funciones Javascript:
```
<script src="UAIProximityEventsV2.js"></script>
```

2. Añadir un controlador para el evento disparado al realizar una detección
```
<script>
 document.addEventListener('ProxEvent', function (e)
        {
 //codigo propio para accionar
        }, false);
</script>
```

   Dentro de este evento es posible refererirse a la cantidad de unidades de tiempo detecatadas mediatne la variable global UAIProxCount.

3. Iniciar la detección mediante la función StartSensing. Esta función recibe tres parámetros:
   - El primero indica si se le da feedback al usuario mediante una vibración cuando se detecta una unidad de tiempo.
   - El segundo indica la cantidad máxima de unidades de tiempo que será detectadas. Cuando se llegue a la última unidad el evento se disparará en forma automática independientemente de si el usuario sigue obstruyendo el sensor de proximidad o no.

- El último es el tiempo en milisegundos de la duración de cada unidad de tiempo. Recomendándose un tiempo de 700 según las nuevas pruebas realizadas, los cuales serán validados finalmente en la siguiente sección del presente artículo.

4. Si se desea dejar de detectar los eventos se debe invocar a la función StopSensing()

VII. VALIDACIÓN DE LA PROPUESTA

Se efectúan pruebas de detección de eventos: 1, 2 y 3 intervalos de tiempo. Organizadas en dos etapas:

- Independencia del Dispositivo: Consiste en efectuar pruebas con distintos usuarios, sobre un mismo equipo. El objetivo de esta prueba es analizar la facilidad de uso de la propuesta y por otro lado adaptar tiempos, en caso de ser necesario. Se analiza si el tiempo en que se configuró la vibración es correcto, evitando de este modo que el tiempo de espera sea demasiado largo (impacientando al usuario) ó corto (impidiendo que pueda mover su dedo antes de generar una siguiente vibración no deseada).
- Pruebas con Distintos Dispositivos: Se prueba en los mismos equipos en los que fue evaluada la API anterior y se analiza si funciona correctamente la detección por parte de los sensores de proximidad.

A. Etapa I – Independencia del Dispositivo

Se generó una página desde la cual es posible configurar el tiempo de detección, esto simplemente ha sido realizado para poder decidir cuál sería el tiempo razonable en el que el usuario debe obstruir el sensor para detectarse un evento y dar como respuesta una vibración. En la figura 6 se presenta la captura de pantalla utilizada para la configuración y pruebas iniciales.

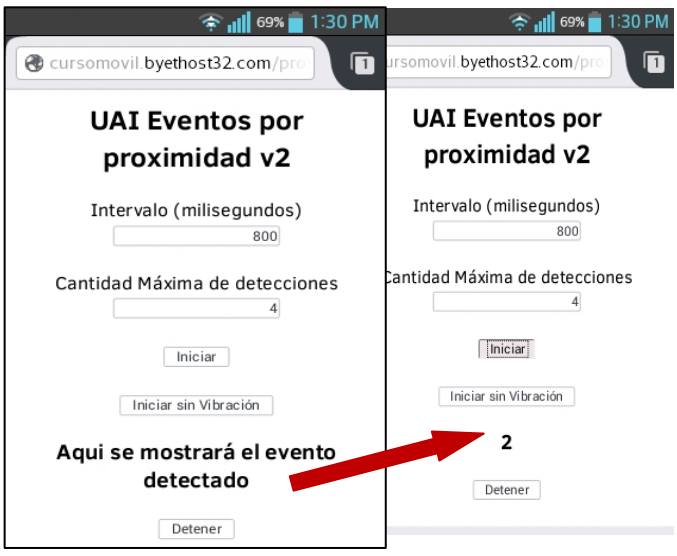

Figura 6. Pantallas de configuración para pruebas.

En esta pantalla se puede setear:

- Intervalo en milisegundos: Es el tiempo de detección del evento.
- Cantidad Máxima de Detecciones: Implicará cuantos eventos habrá configurados.

Como esta misma pantalla se utiliza para probar eventos puede verse debajo el número 2 que muestra el evento detectado.

Las pruebas realizadas se efectuaron con 10 usuarios (independientemente de las pruebas que siempre se efectúan al programar la API en la UAI – Universidad Abierta Interamericana- con el equipo de investigación del CAETI – Centro de Estudios en Tecnología Informática). Esos usuarios han sido divididos en dos grupos, el primer grupo tenía conocimiento del funcionamiento de las pruebas por haber probado la API anterior, se les explicó la diferencia y el segundo grupo eran nuevos potenciales usuarios dispuestos a probar los eventos. Con un máximo de 4 eventos detectables se concentró la prueba en la detección de eventos: 1, 2 y 3. Probándose con distintos intervalos desde 400 a 800 milisegundos. Cada usuario comenzó probando con el tiempo más bajo (que es la mitad del prefijado originalmente). Por cada tiempo se realizaron 4 intentos de detección de cada evento posible.

Cuanto menor es el tiempo prefijado mayor es la posibilidad de error cometido de haber querido retirar el dedo (dejar de obstruir el sensor y no haberlo logrado en el tiempo previsto). Se comenzó en 400 milisegundos subiendo de a 100 milisegundos por cada tiempo de prueba. Al llegar a un valor de tiempo en el cual la detección es perfecta para ese usuario se prueba un tiempo intermedio entre aquel que tuvo errores y el tiempo sin errores en la detección. En la Tabla I se presenta la cantidad de fallos por evento y tiempo resultantes de la prueba con el primer usuario. Este usuario sobre 5 pruebas por cada tiempo obtiene resultados incorrectos con 400 ms, entonces se lo somete a la misma prueba con 500 ms, notándose que la situación mejora pero aún hay errores, se incrementa el tiempo de prueba a 600 como no hay errores en ninguna de las 5 pruebas efectuadas (15 detecciones válidas), se prueba con un tiempo intermedio 550 apareciendo un error.

TABLA I
RESULTADOS DE LAS PRUEBAS EFECTUADAS – USUARIO 1

Tiempos	Eventos		
	1	2	3
400 ms	2	4	1
500 ms	1	1	0
600 ms	0	0	0
550 ms	0	1	0

Es decir que para este usuario en particular el tiempo apropiado es 600 ms. Para comprender la metodología se muestra un caso más en la Tabla II, este segundo usuario también tiene dificultades para la detección con 400 ms de forma tal que comienza a probar otros tiempos con incrementos de a 100 y en 600 ms se detecta aún un error, como en 700 ms la prueba es satisfactoria se efectúa nuevamente otra prueba con 650 ms encontrándose en este caso que no hay errores.

TABLA II
RESULTADOS DE LAS PRUEBAS EFECTUADAS – USUARIO 2

	Eventos		
Tiempos	**1**	**2**	**3**
400 ms	5	4	1
500 ms	3	2	0
600 ms	1	0	0
700 ms	0	0	0
650 ms	0	0	0

En estos dos casos mostrados, a modo de ejemplo, para comprender la magnitud de las pruebas realizadas puede notarse que con 400 ms sobre 5 intentos por cada evento al segundo usuario le fue imposible lograr el objetivo y el primer usuario también tuvo en el evento 2, 4 errores de 5 intentos. Esto hace pensar que el tiempo de prueba es verdaderamente muy bajo.

Cuando estos patrones comienzan a advertirse en la mayor parte de los usuarios que realizan las pruebas es posible tomar noción de la necesidad de regular los tiempos pre-fijados.

B. Etapa II – Pruebas con Distintos Dispositivos.

Una vez establecido el tiempo acorde de detección mediante las pruebas efectuadas en la etapa anterior (700ms), se realiza una nueva prueba sobre distintos dispositivos. Para esto se utilizan los mismos dispositivos que para la API anterior contando con equipos de las siguientes marcas: HTC, LG, SANSUMG. Cabe recordar que quedan excluidos de las pruebas los equipos con sistema operativo IOS y WindowsPhone, por no contar actualmente con la posibilidad de instalar Firefox para poder efectuar las pruebas con el sensor de proximidad. En todos los dispostivos la nueva API funcionó correctamente de modo que permite incluir a aquellos dispositivos que quedaban excluidos en la API anterior por no contar con un sensor de proximidad preciso.

VII. CONCLUSIONES

Este artículo presenta una manera novedosa de utilizar al sensor de proximidad permitiendo a través del mismo mejorar la forma de navegación sobre todo en sitios web.

Con HTML 5 es posible acceder a diversos sensores desde la web, entre ellos al sensor de proximidad, con lo cual fue posible construir una API realizada en Java Script que permite reconocer desde el browser los eventos sobre el sensor de proximidad (medición de obstrucción ó no) permitiendo de este modo generar gestos en el aire que sean reconocidos por la API. También se ha agregado vibración para que el usuario sepa que su gesto ha sido detectado y de este modo darle un feedback. Se definieron inicialmente tres gestos: pase corto, pase largo y doble pase; que al momento de hacer las pruebas pudieron ser ajustados los tiempos de espera en ms (milisegundos). Sin embargo, las variantes de los sensores de proximidad que se encuentran en las distintas marcas de teléfonos celulares condujeron a que esto funcionara en algunos equipos particulares, pero no en forma universal. Se descarta la posibilidad de configurar en algún equipo particular la sensibilidad del sensor, dado que por una parte no está disponible esa opción en todos los dispositivos y en aquellos que si lo está, implica una serie de pasos que son muy dificultosos para un usuario final, entre ellos necesitando acceder a menues ocultos. Fue por eso que se limitó la API para permitir que funcione en todos los dispositivos independientemente de las características del sensor utilizado. La segunda API creada pudo ser testeada con excelentes resultados resolviendo el problema que se generaba inicialmente.

REFERENCIAS

[1] W3C. “Mobile Web Best Practices”. 2008. Disponible en: http://www.w3.org/TR/mobile-bp/
[2] W3C. “Mobile Web Application Best Practices”. 2014 Disponible en: http://www.w3.org/TR/mwabp/
[3] W3C. “MobileOK Cheker”. 2015 Disponible en: https://validator.w3.org/mobile/
[4] W3C. Marckup Validator Service. 2015 Disponible en: https://validator.w3.org/
[5] W3C. CSS Validator Service. 2015. http://www.css-validator.org/
[6] J. Nielsen, R. Budiu “Usabilidad. Book Mobile Usability”. Pearson Education. 2012.
[7] S. Krug, “Don't make me think: A common sense approach to web usability”. Pearson Education India. 2005.
[8] R. A. Rodríguez, P. M. Vera, M. R. Martínez, & L. Verbel de La Cruz, "Aprovechamiento del hardware de los dispositivos móviles para la construcción de nuevas aplicaciones." XVI Workshop de Investigadores en Ciencias de la Computación. 2014.
[9] J. Nielsen “Why You Only Need to Test with 5 Users”. 2000. Disponible en: http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/
[10] N. Bevan, C. Barnum, G. Cockton, J. Nielsen, J. Spool, & D. Wixon, “The magic number 5: is it enough for web testing?.” CHI'03 extended abstracts on Human factors in computing systems, pp. 698-699. 2003.
[11] Deveria “A. Cant I Use Proxi. Proximity API”. 2016. Disponible: http://caniuse.com/#search=proxi

Pablo Martín Vera. Ingeniero en Informática (UNLaM-Universidad Nacional de La Matanza). Doctor en Ciencias Informáticas (UNLP-Universidad Nacional de La Plata). Profesor de grado en el UNLaM y UTN (Universidad Tecnológica Nacional). Docente de posgrado en UNLaM y en la UAI (Universidad Abierta Interamericana). Director de proyectos de investigación en UNLaM y UAI. Supervisor de PPS (Prácticas Profesionales Supervisadas) de alumnos de Ingeniería en UNLaM, Director de Becarios (UAI y UNLaM). Revisor de publicaciones académicas. En el ámbito privado, es Director de Tecnología en una empresa de telecomunicaciones.

Rocío Andrea Rodríguez. Ingeniera en Informática (UNLaM-Universidad Nacional de La Matanza), Doctora en Ciencias Informáticas (UNLP-Universidad Nacional de La Plata). Es docente de grado en la UNLaM y UTN (Universidad Tecnológica Nacional); docente de postgrado en la UAI (Universidad Abierta Interamericana) y UNLaM. Directora Académica del GIDFIS (Grupo de Investigación y Desarrollo en Innovación de Software). Directora de proyectos de investigación en la UAI y UNLaM. Además dirige pasantes, becarios y tesistas. Ha participado como jurado de tesis y revisora de: artículos, proyectos de extensión universitaria y programas co-financiados. Siendo autora de: libros, capítulos de libros y artículos académicos.