PROYECTO FINAL DE INGENIERÍA

MARCO DE REFERENCIA ESTANDARIZADO PARA EL DESARROLLO DE FUNCIONALIDADES EN RESPONSIVE WEB DESIGN

Balcarce, Ignacio Sebastián – LU 132113
Ingeniería en Informática

Devoto, Ariel – LU 122913
Ingeniería en Informática

Tutor/es:
Rubín Aymá, Alejo Fedor

Julio 10, 2015

UNIVERSIDAD ARGENTINA DE LA EMPRESA

FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

Agradecimientos

Habiendo llegado casi al final de nuestro trabajo y al mismo tiempo de nuestra carreara universitaria, sin dudas son muchas las personas a las que deberíamos extenderle nuestro agradecimiento.

Principalmente a nuestro tutor de tesis, quien no sólo nos dio su apoyo para el desarrollo de este tema, sino que además depositó su confianza en que llevaríamos adelante este trabajo con éxito. Por otra parte, queremos extender nuestro agradecimiento hacia él por su tiempo y dedicación, aportes, paciencia y guía a lo largo de estos años en el transcurso de nuestra carrera en UADE.

Por último, agradecer a nuestras familias, quienes nos apoyaron y acompañaron en el transcurso de esta carrera.

Resumen

Al día de la fecha no existe un marco de referencia donde se agrupen y fundamenten distintos estudios sobre interfaces realizados bajo la metodología Responsive Web Design (RWD), así como también el uso apropiado de ciertos tipos de interfaces y sus componentes de acuerdo al caso de uso donde se deseen aplicar.

Este PFI intenta indagar los motivos por los cuales deben adaptarse ciertos estándares a estos diseños a fin de mejorar la experiencia de usuario en el uso diario de ciertas aplicaciones web a través de un dispositivo móvil. En el siguiente trabajo se analizará una serie de funcionalidades seleccionadas, las cuales se presentan en su mayoría en el uso de las aplicaciones que existen en la actualidad.

Para cada funcionalidad se desarrollarán las siguientes etapas de modo iterativo:

- Investigación de funcionalidad: modo de uso, experiencia de usuario, casos de estudio, análisis de ventajas y desventajas.
- Solución a limitaciones y problemas encontrados: estudio de interfaces en las que se intente mejorar y especificar nuevos estándares sobre lineamientos a seguir. Soportar dichas decisiones con los estudios realizados.
- Conclusiones sobre experiencias y observaciones encontradas.

Queda expuesta la implementación de un marco de referencia de funcionalidades y recomendaciones bajo la metodología de RWD para un conjunto de funcionalidades en modo adaptativo.

Abstract

At the time of writing this document, there is no reference from studies that describe the foundation for a framework to work with user interfaces (UI) under the methodology of Responsive Web Design (RWD) in addition to the use cases where it wants to be applied.

This PFI aims to explore the reasons behind on which certain standards should arise in order to improve the user experience (UX) on Web applications.

The following work presents an analysis from selected functionalities, which are commonly used for in applications.

For each of these functionalities the following steps will be developed iteratively:

- Functionality research: ease of use, user experience (UX), case studies with advantages and disadvantages analysis.
- Solutions to encountered challenges and difficulties: UIs study for improvements and new standards with a specification of rules that should be followed. Support these decisions and recommendations with previous findings.
- Conclusions based on experiences and observations.

It is therefore presented a framework with a set of functionalities and recommendations to follow under the RWD methodology for the development of UIs.

Contenidos

Introducción

Objetivos

El objetivo del presente Proyecto Final de Ingeniería (PFI) se encuentra en desarrollar un marco de referencia bajo el concepto de Responsive Web Design (RWD), abarcar un número de funcionalidades utilizadas en la mayoría de las aplicaciones y sistemas actuales, realizar un estudio de cada una de ellas e intentar alcanzar un resultado claro y fundamentado de por qué se logra la solución a plantear en el PFI. Esta solución intenta asumir que las presentes funcionalidades se puedan llegar a utilizar de la misma manera, sin perder la experiencia de usuario, en cualquier tipo de dispositivo móvil del cual se accedan y utilicen.

Estado del arte

Este es un estudio que se encuentra en progreso, ya que hoy en día existen ciertos lineamientos y sugerencias respecto a aspectos puntuales, aunque al día de la fecha no se encuentra aún un marco de referencia como el que se plantea en este PFI. En el mismo, se fundamentan y agrupan distintos estudios con criterios de diseño de interfaces para distintos tipos de plataformas y dispositivos para poder clasificar la información.

Propuesta de solución

Este PFI intenta indagar los motivos por los cuales deben adaptarse ciertos estándares a estos diseños a fin de mejorar la experiencia de usuario en el uso diario de ciertas aplicaciones y servicios desde un dispositivo móvil. Este marco de referencia aspira a ser el inicio de un trabajo sobre el cual pueda a futuro generarse un estudio completo sobre la materia que permita estrechar esa diferencia que existe actualmente en la mayoría de los sistemas y aplicaciones que dificultan que los usuarios puedan lograr la misma facilidad de uso accediendo desde dispositivos móviles, permitiendo así que en poco tiempo estas diferencias ya no existan y que decisiones como estas sean tomadas en consideración desde el inicio del diseño de sistemas como otros aspectos habitualmente considerados.

Estructura del informe

En la sección “Tablas de datos tabulados” se presenta al lector una serie de recomendaciones a tener en cuanta al momento de utilizar tablas de datos tabulados.

En la sección “Menú” se presentan al lector los distintos tipos de menú más utilizados hoy en día y se extiende la definición de ellos sobre recomendaciones que indican en qué casos debería utilizarse cada tipo de menú, con sus respectivos alcances.

En la sección “Imágenes” se presenta al lector una serie de soluciones desde cómo brindar imágenes desde distintas fuentes o técnicas se puede lograr resultados similares bajo diferentes métodos.

En la sección “Mapas” se presentan al lector distintas maneras de uso de mapas como integración de aplicaciones nativas en dispositivos móviles, así como también adaptación en pantalla y uso de mapas estáticos.

En la sección “Formularios” se presentan al lector las funciones disponibles para poder crear formularios que mejoren la experiencia de usuario por medio de adaptabilidad, distintos tipos de validaciones y campos de ayuda, como así también el uso de controles útiles.

En la sección “Videos” se presenta al lector una serie de soluciones desde cómo brindar videos desde distintas fuentes o técnicas logrando resultados similares bajo distintos métodos.

Tablas de datos tabulados

Introducción

Las tablas de datos tabulados son uno de los principales elementos que posee una aplicación web para mostrar información de manera ordenada. Para que sean parte de un diseño *RWD*, es necesario que las mismas sean adaptables frente a cualquier tamaño de pantalla o dispositivo que se presente. El problema al cual uno se enfrenta en caso de no contemplar este diseño se puede demostrar en el siguiente ejemplo, donde simulamos una aplicación de contactos de un usuario:

Figura 1: Ejemplo nº 1 de css-tricks.com de vista incómoda de tablas.

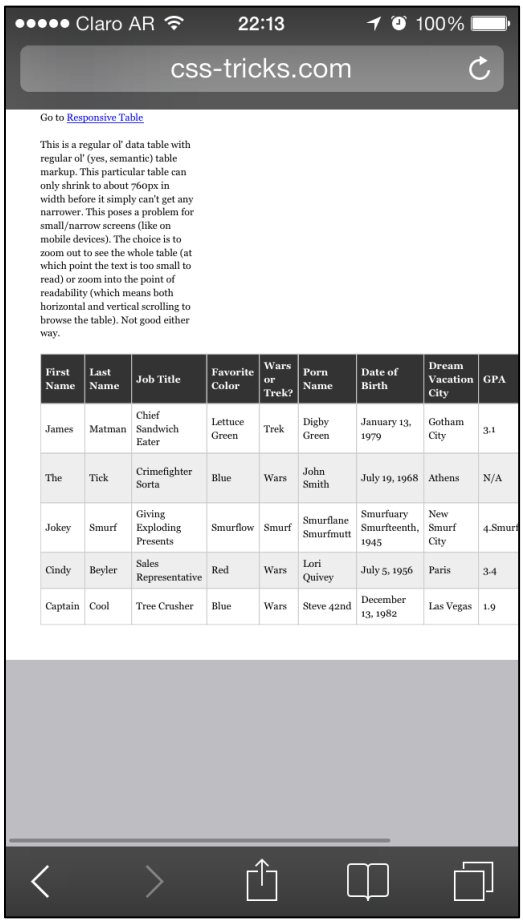


Figura 2: Ejemplo nº 2 de css-tricks.com de vista incómoda de tablas.

En el primer ejemplo podemos denotar que los contactos se encuentran cómodamente presentados ya que los mismos están en su tamaño original, pero se dificulta la lectura frente a tablas amplias en columnas y filas.

En el segundo ejemplo, la tabla entera se visualiza en la pantalla, pero la lectura resulta más compleja. Debido a esto, deberá utilizarse para ello funciones de zoom y se volvería a la situación de la figura número 1. Esto no ocurre únicamente en dispositivos móviles sino también en pantallas de computadoras de escritorio antiguas, donde la resolución no es de las más utilizadas actualmente.

La idea a partir de esto es poder unir ambas opciones de modo tal de utilizar sus beneficios. Para ellos utilizaremos los principios de *RWD* y estableceremos reglas de modo tal que el contenido se cargará de manera dinámica frente a parámetros de tamaño de pantalla, pudiendo así ver el contenido en un dispositivo móvil de la siguiente forma presentada en la figura 3.

Figura 3: Ejemplo de css-tricks.com de una tabla bien ajustada a la lectura móvil.

En este ejemplo podemos ver como mediante un poco más de personalización, se pueden lograr técnicas para dar mayor facilidad cuando se tienen que mostrar tablas de muchas columnas en un dispositivo de manera responsiva, en donde cuanto más pequeño sea el espacio en donde deberá mostrarse, menor será la cantidad de columnas a mostrar. La idea principal es la de brindarle al usuario la posibilidad de elegir las columnas que desea visualizar.

Cabe destacar que el ejemplo citado no es exclusivo y que lo demostrado es sencillamente una decisión de diseño. Pueden existir diversas formas de mostrar información en lugares acotados, siendo esta una de ellas.

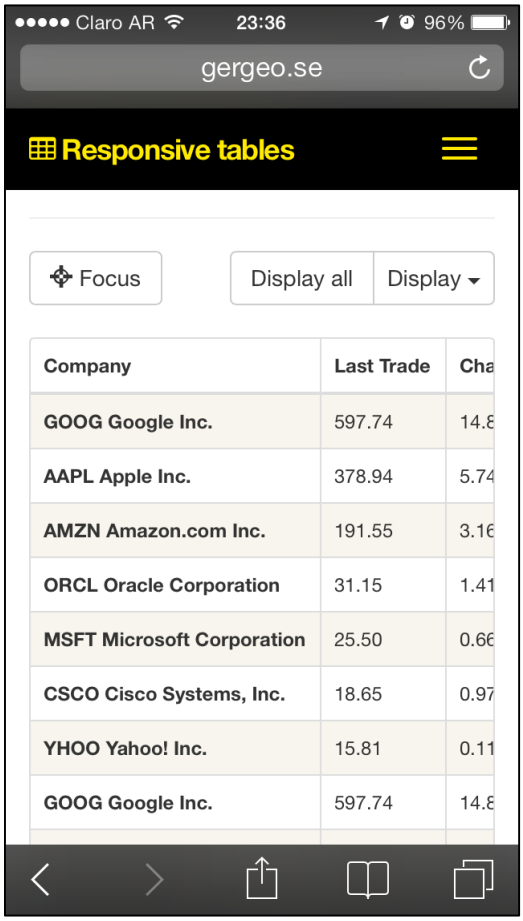


Figura 4: Ejemplo nº1 de gergeo.se de tablas dinámicas

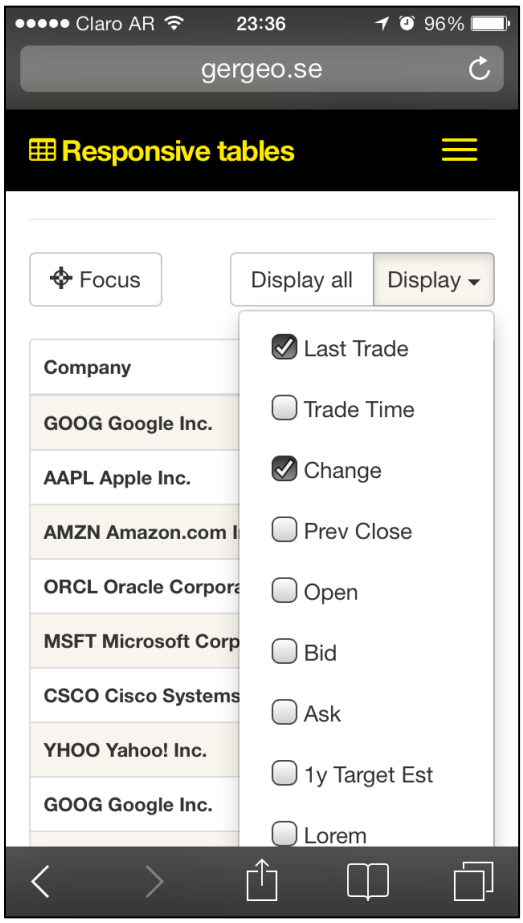


Figura 5: Ejemplo nº2 de gergeo.se de tablas dinámicas

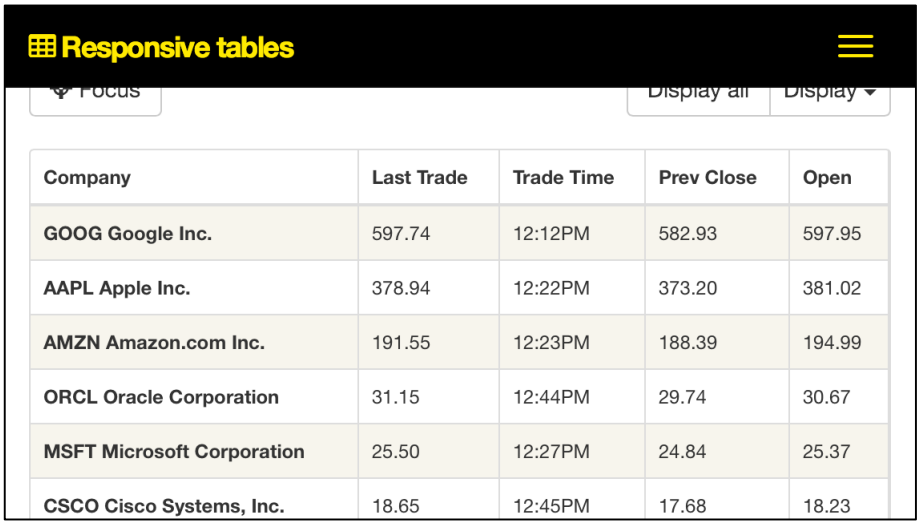


Figura 6:Ejemplo nº3 de gergeo.se de tablas dinámicas

Siguiendo con las referencias a distintas aplicaciones de organización del usuario, sucede lo mismo con los eventos en los calendarios. Pensando en el objeto básico en realidad estamos hablando de una tabla, pero al referirse a una aplicación de este tipo, no puede tener un tamaño fijo, y las vistas que tiene deben variar en cuanto a su complejidad frente al tipo de usuario que este navegándolo. Lo que se intenta realizar es no caer en un ejemplo del estilo:

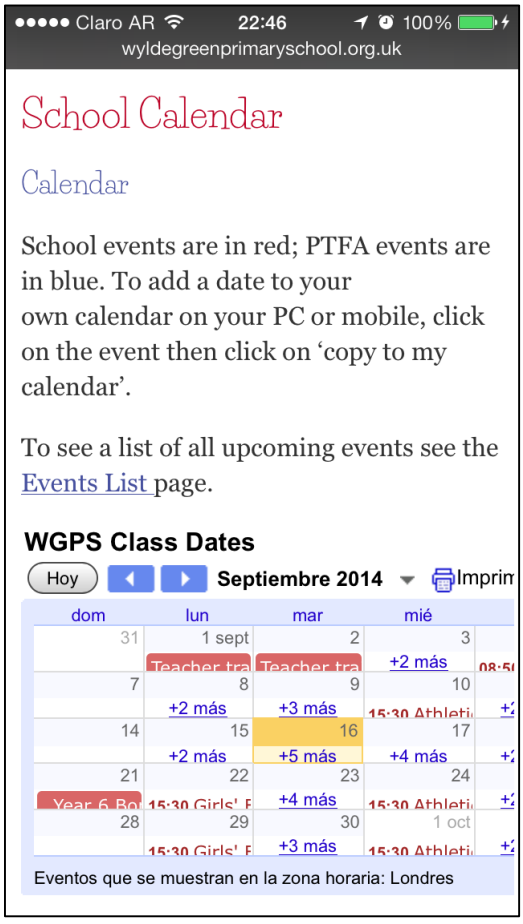
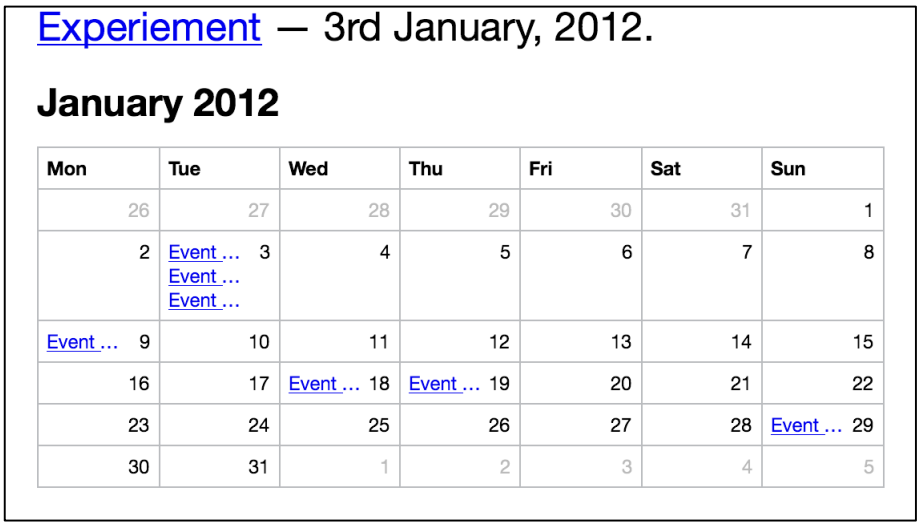


Figura 7: Ejemplo de calendario no conveniente.

En un ejemplo de calendario bien implementado en dbushell.com podemos observar lo que se considera un estándar conveniente, donde en vistas grandes, el calendario se muestra en forma de tabla pero en las vistas más pequeñas los eventos se ordenan en una única columna, permitiendo la lectura en una vista agradable.



Experiement — 3rd January, 2012.

January 2012

Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	29	30	31	1
2	Event … 3 Event … Event …	4	5	6	7	8
Event … 9	10	11	12	13	14	15
16	17	Event … 18	Event … 19	20	21	22
23	24	25	26	27	28	Event … 29
30	31	1	2	3	4	5

Figura 8: Ejemplo nº1 de un calendario conveniente.

Figura 9: Ejemplo nº2 de un calendario conveniente.

Conclusiones

No existen mayores problemas para poder adaptar este tipo de elemento en una aplicación web. Bastará con aplicar los conceptos básicos de HTML5 y CSS3 sobre ellos para obtener los resultados expuestos anteriormente. Los principales obstáculos que presenta el tema evaluado en el presente documento se presentan en las siguientes secciones.

Menú

En la mayoría de las aplicaciones actuales, así como también en páginas web, es condición necesaria poder separar la información que se quiere brindar al usuario en distintas secciones. Asimismo, para que el usuario tenga una experiencia grata a la hora de navegar, las mismas no sólo deben ser entendibles en su contexto, sino también en su contenido. Del mismo modo, se deberá contemplar que el usuario puede estar navegando desde cualquier tipo de dispositivo y que su experiencia de usuario no debe verse afectada por esto.

Esto definirá nuestro primer requisito a cumplir a la hora de desarrollar un menú contextual claro y compatible sobre un dispositivo móvil.

¿De qué forma se podrá cumplir con tal condición? Se analizará a continuación diversas metodologías utilizadas actualmente, comparando sus ventajas y desventajas y así poder llegar a una solución correspondiente a las mejores prácticas, utilizando las ideas que se consideren óptimas de cada una, siempre que se considere viable.

Slide-out navigation / Left-hand navigation

Left-hand navigation o *Slide-out*, es un menú horizontal que despliega una lista de opciones cuando se selecciona uno de los elementos principales de la pantalla. Al seleccionar dicho elemento se presenta por debajo una tabla que sirve de acceso a distintas opciones. Esta tabla se encuentra originalmente oculta de la página principal. Una vez presentado el menú, seleccionando cualquiera de estas opciones, traerá nuevamente a pantalla la vista principal con la opción seleccionada. Para volver a la pantalla principal, se debe seleccionar la parte que mantiene presente en vista en una pequeña porción, de la vista principal o simplemente arrastrando la misma a su lugar de origen en caso de encontrarse trabajando con dispositivos móviles de tecnología táctil.

Casos de estudio

Algunas redes sociales también suelen implementar el uso de navegación *Slide-out* para el acceso a sus funciones. Por ejemplo, en las webs de LinkedIn y Unwire, encontramos este tipo de menú, en el cual se mantiene una baja cantidad de elementos, mostrando entre ellos distintos tipos de acceso identificados por palabras y símbolos.

En cuanto a Unwire, la interfaz que se presenta es bastante simple, con su cantidad reducida de elementos en su menú y una apertura proporcionada de su pantalla principal debido a ello. No presenta riesgo alguno de deficiencia en la navegación debido a dicha simplicidad.

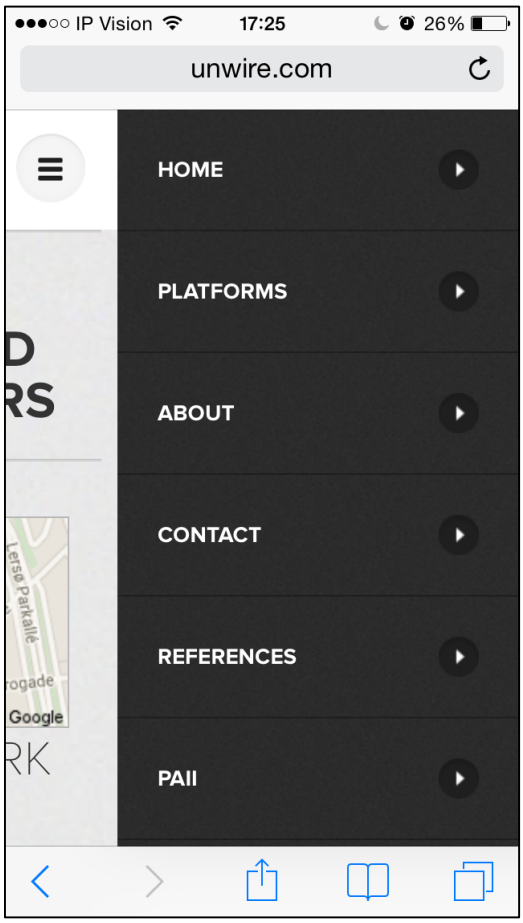


Figura 10: Menú Slide-out en Unwire.

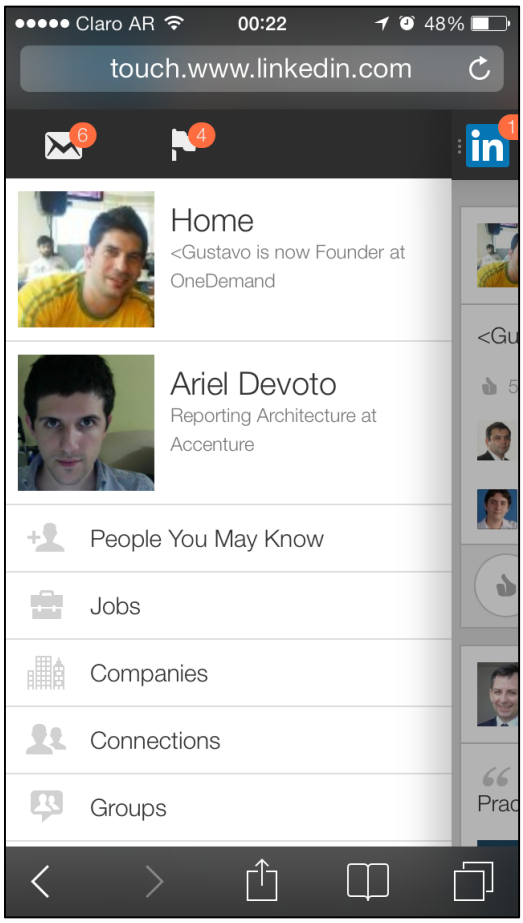


Figura 11: Menú Slide-out en Linkedin.

En cuanto a la web de LinkedIn, el funcionamiento consiste en seleccionar el icono del menú posicionado en la parte superior izquierda de la pantalla. A continuación, la pantalla principal se desplaza hacia la derecha, dando lugar al menú el cual ocupa casi el 80% de la pantalla, quedando así el 20% restante en uso para mostrar el contenido de la pantalla principal. De esta manera, se logra dar al usuario conocimiento de que lo que estaba utilizando, el mismo contenido no se perdió y aún existe como punto de referencia de donde se encuentra navegando. También existen variantes según el ancho que se desea ocupar dependiendo de las necesidades.

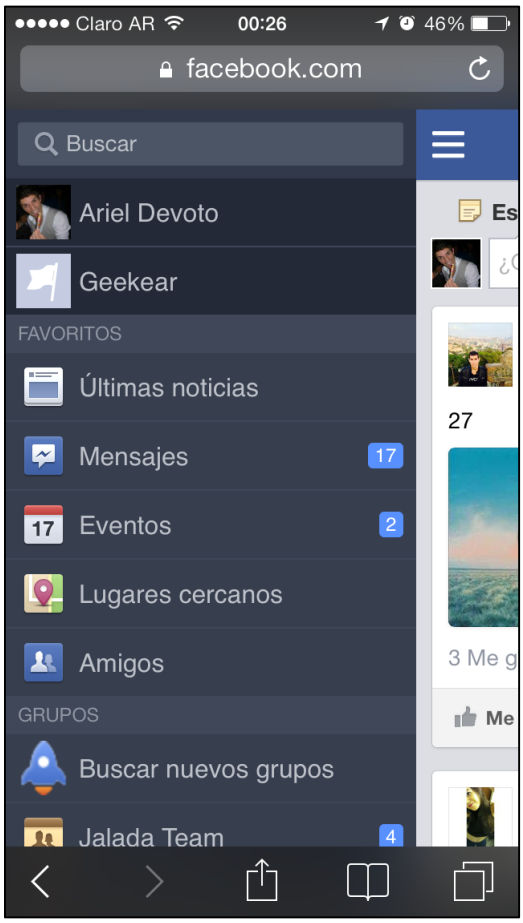


Figura 12: Menú Slide-out en Facebook.

Seguidamente, en la web de Facebook se puede encontrar otro punto a favor (aunque la imagen no lo demuestre) en el cual existe una extensión que tiene el menú, teniendo una gran cantidad de elementos de diversa índole; elementos con visor de cantidad de eventos recibidos, elementos que abrirán un próximo nivel de menú, una barra de búsqueda entre otros.

Ventajas

Ciertamente, su mejor uso se vio en aplicaciones o webs de contenido relacionado (aplicaciones orientadas al contenido), especialmente cuando existen muchas vistas que pueden crecer dinámicamente. De esta manera, se logra presentar de manera intuitiva muchas funciones en una única vista. Sobre todo, provee suficiente espacio para almacenar textos largos, llegando a ocupar casi un 80% del ancho de la pantalla.

Al mismo tiempo que la interfaz de usuario se vuelve más comprensiva y amigable, también permite integrar una extensa cantidad de componentes como búsqueda, barras de progreso, controles, vista previa de imágenes y video así como también accesos con distintos tipos de iconos o imágenes. También es posible ofrecer al usuario la opción de agregar atajos a otras partes de la aplicación que desee y el sistema le permita.

La escalabilidad es un factor importante y debido a la posibilidad de extender dicho menú a cualquier cantidad de elementos que uno requiera sin afectar la navegación, existen ciertas consideraciones de agrupación que deben tenerse en cuenta.

Desventajas

Suponiendo que exista una gran cantidad de elementos a mostrar dentro del menú, es importante considerar que mostrar en la pantalla muchos de ellos podría resultar en una navegación compleja y confusa. A causa de ello, es apropiado mantener el agrupamiento de las opciones ordenadas, así como también incluir menús anidados dentro de dichas opciones.

Debe tenerse en cuenta que la escalabilidad en relación a la cantidad de elementos existentes en el menú puede traer aparejado el problema de una carga lenta de navegación de la web o aplicación. En consecuencia, esto traería problemas a la experiencia de usuario tratándose de dispositivos móviles.

Drop-down

El menú *Drop-down* suele presentarse por medio de un botón o icono de acceso, el cual al seleccionarse despliega este menú hacia abajo presentando el resto de las opciones que originalmente se encontraban escondidas. Las opciones de diseño varían indicando una flecha hacia abajo para ayudar al usuario a identificar que aquel botón o icono es un menú que se extenderá en esa dirección, así como también flechas hacia arriba y abajo. Este menú combina la capacidad de una lista *Drop-down* / *Pop-up*.

Al seleccionarse alguna de las opciones en este menú, el mismo se contrae nuevamente a su estado original presentando como título en la parte superior de la pantalla aquella opción que se había seleccionado.

Casos de Estudio

Al analizar este caso en la web de Facebook, se puede observar la simplicidad de la interfaz y como dicha transición hace que no afecte o confunda al usuario en cuanto a dónde se encuentra situado. El contenido desplazado no se pierde y pueden encontrarse todas las opciones pertenecientes al mismo, si el usuario continua navegando hasta el fin del menú.

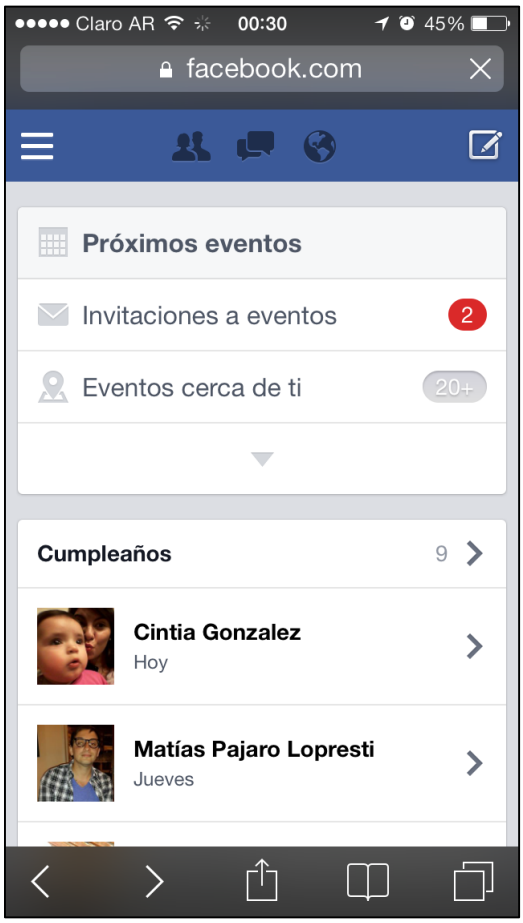


Figura 13: Menú drop-down de Facebook cerrado.

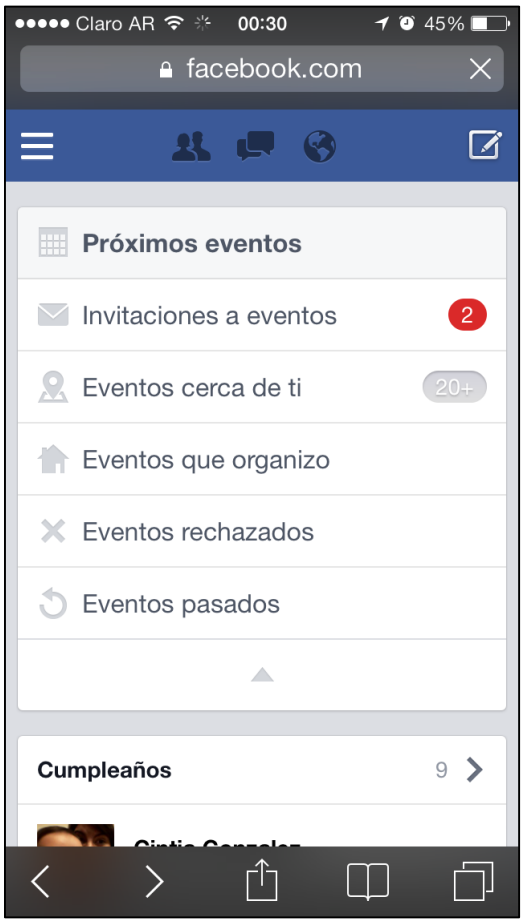


Figura 14: Menú drop-down de Facebook abierto.

En contraste, puede notarse fácilmente la dificultad que existe ante la necesidad de tener que agregar un gran número de ítems a este menú. En consecuencia, puede decirse que este menú no tiene una gran escalabilidad luego de un número limitado de elementos. Aun así, este menú permite el agregado de información adicional dentro de los mismos ítems del menú, tales como una imagen que referencie a la sección que pertenece, así también un contador de notificaciones o referencias sobre cada una de ellas, sin por esto perder simplicidad en su totalidad y pudiendo así mejorar la experiencia de usuario.

Teniendo en cuenta otro caso de estudio como la web de Deepak Chopra, tenemos nuestro primer caso de estudio en el cual el menú *drop-down* es incorrecto desde el punto de vista de *RWD*; El diseño parece no tener problemas con la utilización de dispositivos que no sean táctiles. Sin embargo, el problema surge en dispositivos móviles con tecnología táctil ya que el tamaño de dicho menú no aumenta en proporción al tamaño del dispositivo en

el que se esté visualizando, haciendo que la navegación sea complicada y difícil de seleccionar. Es extremadamente importante considerar que este problema puede aplicarse no solo a *drop-down* menú, sino a cualquier tipo de menú frente al análisis de *RWD*.

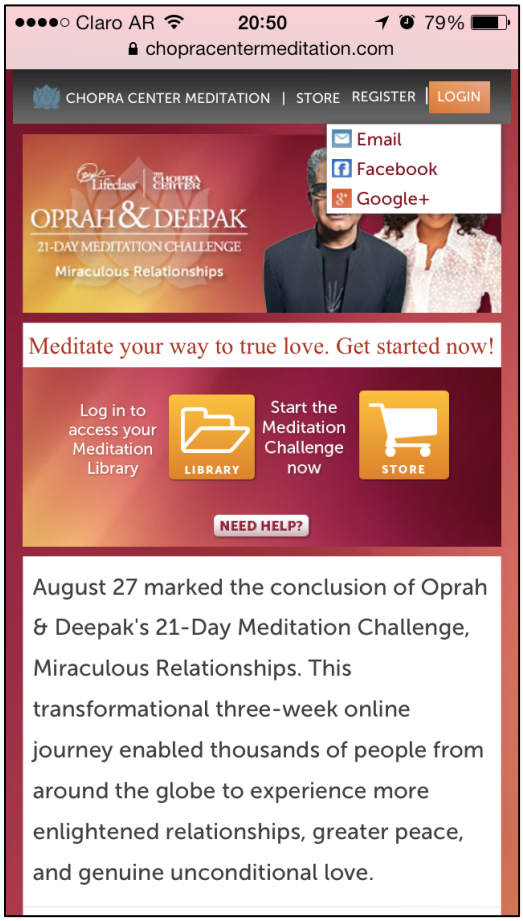


Figura 15: Menú drop-down en Chopra Meditation Center.

Ventajas

Se destacan por su gran utilidad menús posicionados en la parte superior de la página aunque estos menús también pueden ser incluidos en otras partes del contenido. En conjunto con menús en la parte superior. Siempre y cuando se mantenga simple su diseño, el uso de espacio para mostrar el mismo será mínimo, dejando el contenido correspondiente a dicho elemento por debajo del menú. Esto facilita la navegación y posición del usuario al poder determinar en dónde se encuentra situado en la página o aplicación. Este menú resulta

intuitivo y familiar para personas con poco conocimiento de tecnología debido a uso similar en el diseño de páginas webs para desktops.

Desventajas

Como se mencionó anteriormente, la escalabilidad en estos menús es limitada debido a que su tamaño no debería superar la altura de la misma pantalla del dispositivo que se esté utilizando. El desplazamiento tampoco aplicaría como solución. Lo mismo ocurre ante la posibilidad de mostrar sus elementos en formato cascada; al ser uno el tamaño de la pantalla un factor limitante, no es posible realizar sub-menús ya que en unos pocos niveles el menú estaría fuera de la pantalla.

Tabs

Este tipo de menú sigue una estructura de separación en distintos tipos de paneles o secciones. El usuario accede a cada uno de ellos seleccionando el icono respectivo a la sección. Se lo utiliza frecuentemente en aplicaciones de tipo modular, donde el contenido de una sección no necesariamente se encuentra relacionado con el de otras secciones.

Casos de estudio

El funcionamiento consiste en seleccionar alguno de los iconos que se encuentran en dicho menú, que presentarán el acceso a las pantallas correspondientes a cada uno de ellos. Al ser pocas opciones, debido al espacio disponible, ya sea por el ancho o alto de la pantalla, éstas suelen ser el total soportado por la aplicación. Presenta ventajas al encontrarse estas funciones a la vista de manera que el usuario no tiene que acceder a ellas por medio de atajos presentados en sub-funciones.

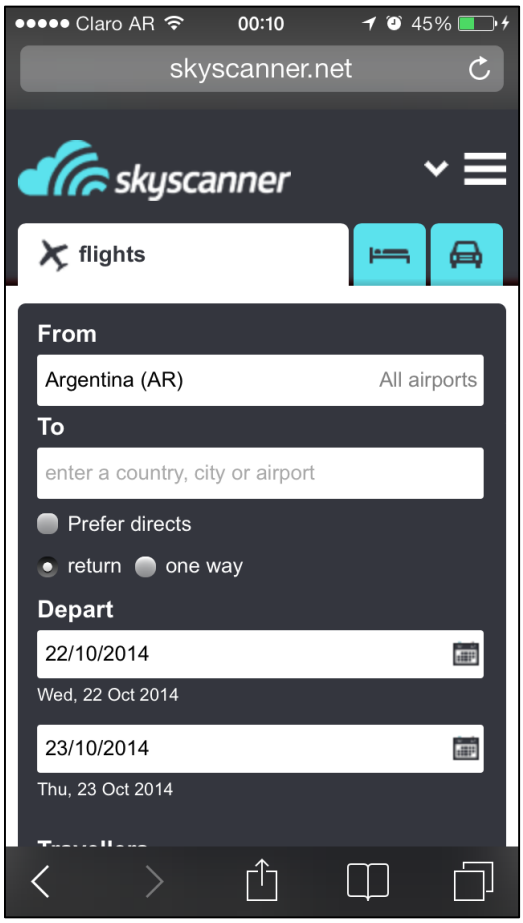


Figura 16 : Ejemplo tab-menu en Skyscanner.net.

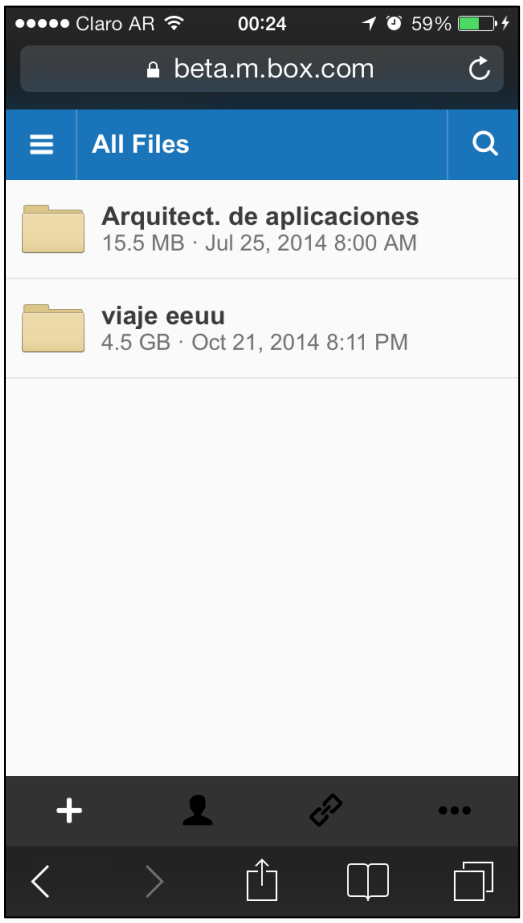


Figura 17: Ejemplo tab-menu en Box.com.

Como podemos observar, en la aplicación de Skyscanner tenemos un menú principal mostrado en modo tabular. Tenemos allí las opciones principales donde el usuario debería interactuar primero; no obstante el usuario, aun avanzando dentro de una de ellas, podrá tener acceso a las otras siempre y cuando lo considere necesario. De esa forma se le da un usuario una sensación de ubicación dentro del flujo de navegación que él realice.

Por otro lado, en la aplicación de Box observamos que el menú tabular se utiliza de otra manera; las opciones allí disponibles son respecto a las acciones que el usuario puede realizar en cualquier momento, relacionadas a sus carpetas y archivos (por ejemplo: Agregar carpeta, Compartir, Propiedades).

Figura 18: Ejemplo de implementación de tabs en tympanus.net.

Ventajas

El mismo espacio utilizado para mostrar una única sección se utiliza para mostrar el resto de las secciones. Permite la segmentación de información de forma compacta, pudiendo ahorrar gran espacio.

Estas secciones modulares son convenientes siempre para mostrar información resumida y de fácil lectura, tales como gráficos, listas y todo tipo de formas visuales simples.

Buenas prácticas en el uso de los menús tabulares se presentan con iconos claros y descriptivos de las funciones que proveen, permitiendo así agrupar más opciones en la barra tabular y brindando una experiencia de usuario más amigable e intuitiva.

Desventajas

Su número limitado de botones, restringido al ancho o alto de la pantalla, no resulta conveniente para múltiples accesos inmediatos, en caso de ser necesitados. Proveer más de una fila de secciones también daría lugar a confusiones, tales como una idea de jerarquía entre las secciones mostradas en la fila superior sobre la inferior.

Mostrar información en distintas secciones que se encuentren relacionadas afectaría la experiencia de usuario, dado que las secciones deben ser completamente independientes para que la navegación sea eficiente.

Conclusiones

Dependiendo del alcance y el diseño que quiera implementarse, pueden utilizarse alguno o una combinación de los menús previamente mencionados. Partiendo desde Facebook, donde se encuentran disponibles una cantidad extensa de funciones, algunas de las cuales crecen dinámicamente puede observarse que se implementan en un mismo lugar el menú tabular, *drop-down* y *slide-out*.

Es importante mencionar que la elección de uno o la combinación de distintos menús, siempre debe considerarse en relación con la usabilidad y experiencia que obtendrá el usuario de ellos. En los casos expuestos se presentan opciones en las que se destacan ventajas como menús dinámicos o de gran número de funciones y sub-funciones (*slide-out*), visibilidad de una cantidad de opciones limitadas al ancho de la pantalla (menú tabular) y un menú extensible que al seleccionarse presenta, en una posición estática en la pantalla, todas sus funciones, para luego contraerse nuevamente al ingresar a alguna de sus opciones (*drop-down*).

Imágenes

Imágenes en RWD contiene, bajo sus posibilidades, numerosas soluciones que pueden utilizarse. Estas mismas se pueden clasificar en dos grupos: Aquellas que resuelven cuestiones desde el lado del cliente utilizando los recursos del mismo, o aquellas en las que se realiza un trabajo del lado del servidor.

Entre aquellas cuestiones más comunes a resolver se encuentran los siguientes: Inicialmente, la variedad de dispositivos en el mercado que capturan imágenes desde 640px hasta un tamaño superior a 3000px, presentan cierta dificultad al momento de asignar un tamaño fijo a una página web o aplicación. Para poder resolver este problema, es necesario que dicha solución sea capaz de adaptar sus imágenes para incorporarse al tamaño de pantalla que se utilice, presentando así una solución eficiente.

Así también, existe una relación directa con la resolución de las imágenes y viene consigo el tamaño de las mismas. Otro factor importante a considerar es la velocidad de conexión de dispositivos móviles, que es inferior a la de las computadoras de escritorio.

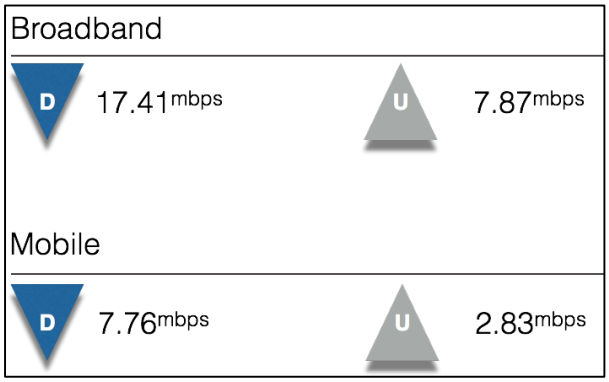


Figura 19: Velocidad por cable vs mobile - Abril 2014.

Por último, existen ocasiones en las cuales reducir una imagen grande conlleva a una desproporción de la misma dando como resultado una imagen distinta a la que se intenta presentar. Para ello se deberá ajustar la imagen a que se reduzca en una cierta proporción, pudiendo ajustar algún punto de enfoque para que la imagen no pierda el objeto de la misma.

Esto mismo se presenta en su caso inverso, en el cual al agrandar una imagen no conlleve a desalinearla o que se pierda la calidad de la misma por cambiar sus propiedades de tamaño.

A continuación se analizara las distintas posibilidades que existen de solucionar los problemas expuestos, y dejar planteados otros que aún no han podido ser resueltos completamente con una solución lo suficientemente razonable desde el punto de vista de las sugerencias y bases en las cuales se plantea este trabajo.

Cropping images

Este método consiste en la ampliación y reducción de la imagen mostrada en pantalla. De este modo siempre se encuentra en proporción con la pantalla en la que se la está visualizando, sin perder detalles de aspecto y calidad.

Caso de estudio

Con la utilización del *focal point*, se evita el problema de perder información al mostrar imágenes para distintos tamaños de pantalla. En el siguiente ejemplo se puede observar cómo es posible que el navegador pueda reducir la imagen, dejándola fuera de los límites de la pantalla y por ende el usuario no podrá verificar de que se trata.

Figura 20: Imagen original a ejemplificar.

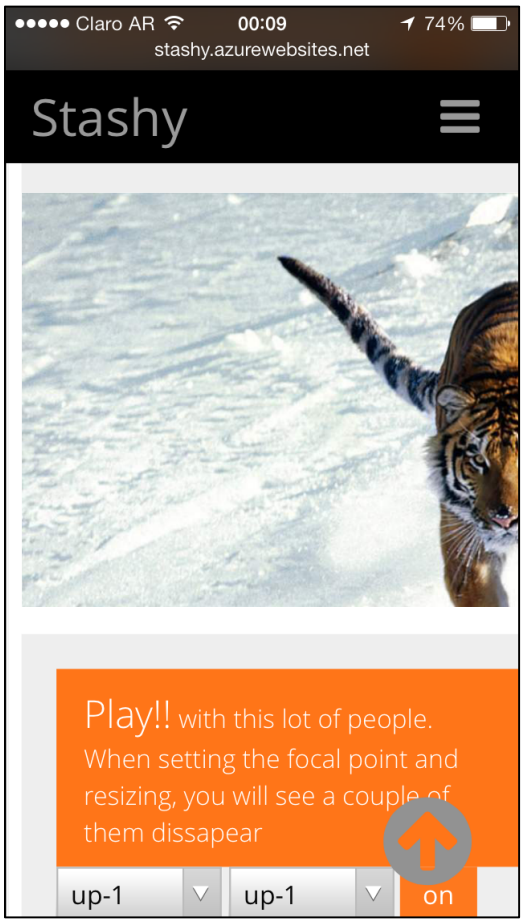


Figura 21: Imagen ajustada sin *focal point*.

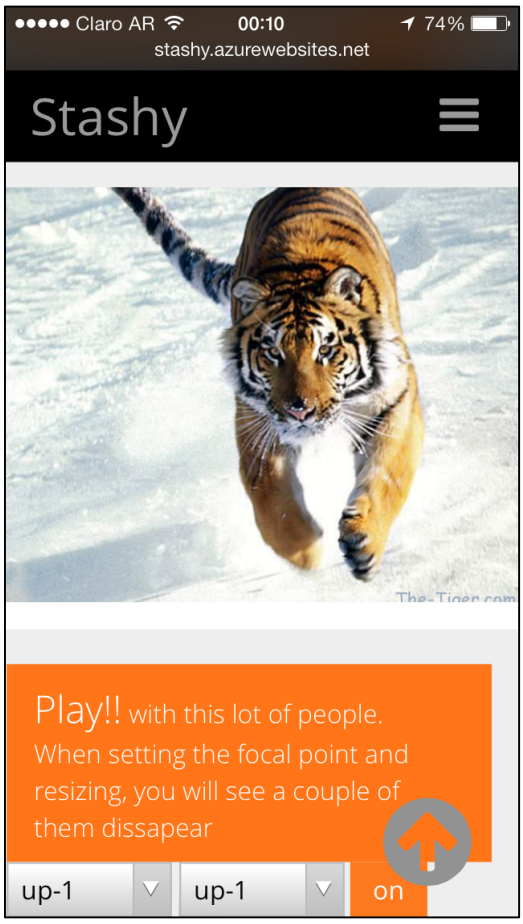


Figura 22: Imagen ajustada con *focal point.*

Ventajas

No requiere ningún tipo de input de parte del usuario. No depende del tipo de tecnología a utilizar. Es parte de los estándares provistos por el lenguaje utilizado.

Existen diversos parámetros adicionales que pueden mejorar en gran medida la muestra de imágenes, ya que por defecto los navegadores al ampliar o reducir una imagen, lo realizan desde una esquina. Dichas variantes pueden ser la reducción en proporción desde todos sus lados. También existen soluciones particulares como el *focal point* donde se puede determinar un punto central en el cual nunca se pierda el ajuste.

Desventajas

Es posible que la escalabilidad no sea 100% segura. Debido a la evolución de los dispositivos móviles y sus pantallas de tamaño superior, es necesario proveer imágenes

que acompañen dicha tendencia. Proveer a un dispositivo pequeño una imagen superior a su resolución daría consecuencias graves al usuario en cuanto a la velocidad, que demoraría en cargarlas, y lo poco eficiente que su navegación resultaría. Inversamente lo mismo sucede en su caso contrario, donde un dispositivo de gran tecnología y resolución disponga de imágenes de baja calidad.

Un problema derivado del caso anterior es el uso del procesamiento del dispositivo, el cual es requerido por los navegadores para ajustar el tamaño de las imágenes. Un dispositivo antiguo puede resultar inutilizable durante varios segundos mientras el navegador intenta realizar estos ajustes.

Grill images

Este método surge como solución a uno de los problemas mencionados en el caso anterior de *Cropping images*. En lugar incorporar una única imagen a mostrar y en base a ella hacer las correcciones necesarias, se provee una serie de imágenes dependiendo del tamaño del dispositivo que se esté utilizando.

Caso de estudio

En las siguientes imágenes, podemos ver una comparación entre la vista en un iPhone 5 y un iPad mini. De ellas se puede observar que las interfaces están organizadas de distinta forma pese a ser la misma web, pero eso no determina nuestro enfoque. Lo que no se ve en las imágenes es que la imagen provista en el IPhone 5 es una imagen mediana para tablets, mientras que en el Ipad mini se muestra una imagen de superior calidad.

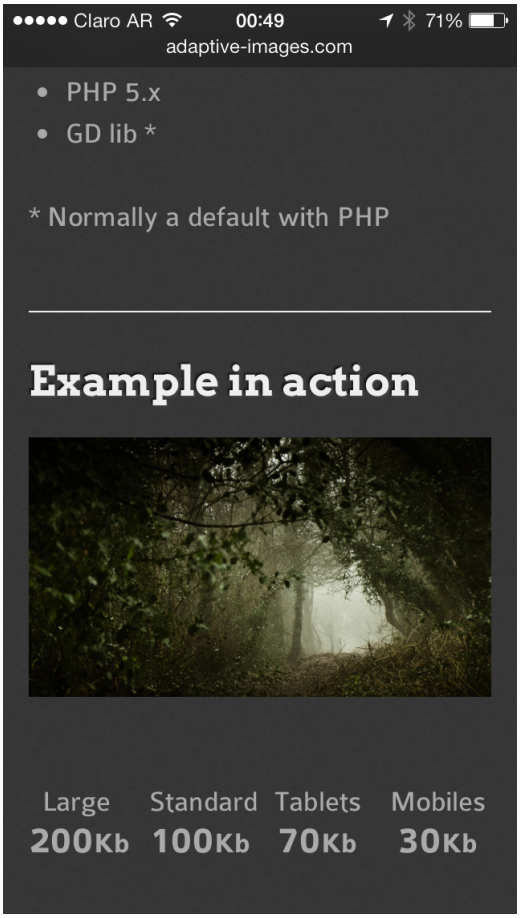


Figura 23: Ejemplo de elección de una imagen en miniatura según vista iPhone.

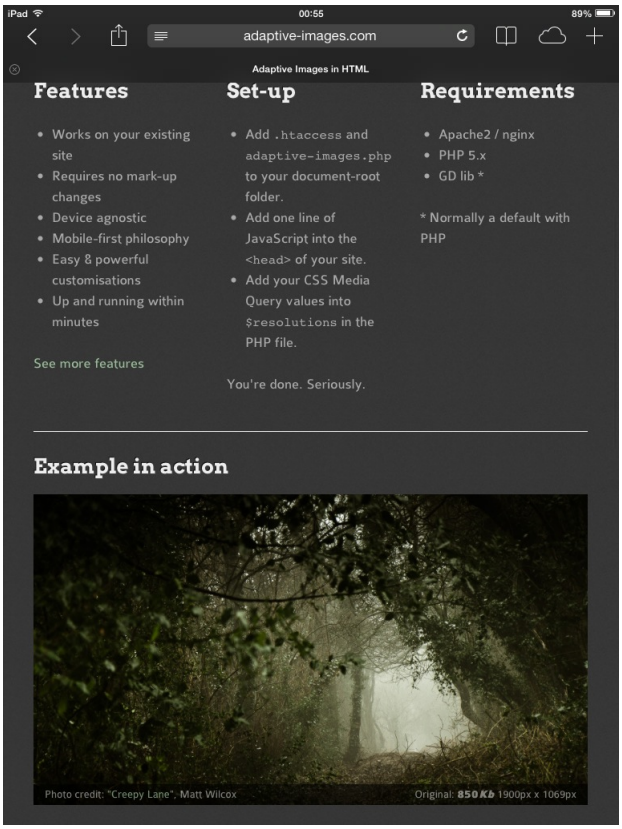


Figura 24: Elección de una imagen en tamaño medio según vista iPad.

Ventajas

Actualmente, con la provisión de 3 tamaños de imagen se logra abarcar la totalidad de dispositivos existentes en el mercado. La escalabilidad hacia el futuro es prácticamente instantánea y requeriría poco esfuerzo.

Al momento de distinguir el tipo de pantalla, se puede también deducir el tipo de conexión que el usuario tiene en el momento. Esto permitiría proveer imágenes *retina display* a aquellos dispositivos que lo requieran. Este ha sido un reciente cambio evolutivo en algunos dispositivos que poseen una considerable mayor cantidad de pixeles por área.

Desventajas

Al proveer distintas imágenes según distintas condiciones, presenta el problema de que todas estas serán cargadas por el navegador. En otras palabras, no importa el dispositivo que se utilice, siempre se tendrá una vista adecuada, pero esto tendrá como consecuencia el tiempo de carga de todas éstas.

El mantenimiento puede tornarse complejo si no se administra de manera correcta.

Puede darse el caso en el que las tres variables presentadas (tamaño de dispositivo, velocidad de conexión, pixeles por área) no estén relacionados, es decir que un dispositivo puede ser pequeño, pero disponga de una gran cantidad de pixeles por área y una baja conexión, por lo cual se le proveerá una imagen de gran calidad, que tardara en cargarse por su conexión.

Soluciones del lado del servidor

Existen varias soluciones que pueden darse del lado del servidor. Entre las más conocidas ya desarrolladas están PictureFill o FlexSlider. La mayoría son métodos para proveer distintas imágenes, corroborando previamente mediante un lenguaje de programación como Javascript alguna especificación del dispositivo del usuario, pudiendo así determinar la elección a la que corresponda.

Ventajas

Se puede determinar desde el comienzo de la carga de página, cuáles serán los recursos más convenientes para cada tipo de usuario. Es de fácil implementación.

Desventajas

Si el usuario carga la página por primera vez y luego quiere cambiar el ancho de la misma, las imágenes cargadas inicialmente permanecerán en el estado inicial en el que fueron cargadas.

La cantidad de imágenes que deben administrarse presenta una relación directa con el costo de mantenimiento y carga de la web.

Conclusión

El análisis realizado ha demostrado que es posible utilizar una solución en base *grill images* para proveer una serie de recursos, la cual será destinada a un tipo de dispositivo y tamaño de pantalla, sin producir una lentitud en su carga. Esto se realizara evitando que un determinado dispositivo deba procesar las imágenes que no están destinadas a ella.

Para ello existe un recurso de CSS3 llamado *media queries*. Éstas nos permiten administrar recursos que deban procesarse en un tipo de pantalla sin la necesidad de tener que cargar otros. Dicha aplicación puede observarse al comparar las figuras 25 y 26 en las cuales a simple vista pareciese que fueran dos sitios totalmente distintos cuando en realidad es el mismo, solo que frente a resoluciones de pantalla distintas obtendremos un diseño u otro.

Figura 25: Tamaño grande de la web Zara (+1024 píxeles).

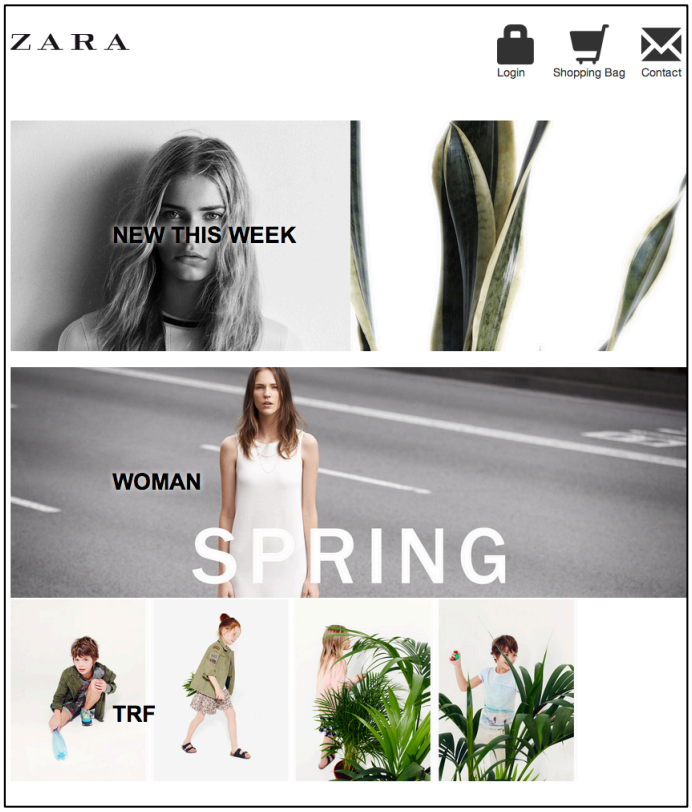


Figura 26: Tamaño pequeño de la web Zara (menos de 1024 píxeles).

No obstante, es posible también aplicar a la par el método de *cropping images*, el cual dará al navegador la tarea de ajustar las imágenes para que coincidan con su ancho y alto de la pantalla, según se especifique.

Esta solución es puramente realizada en HTML y CSS, por lo cual no requerirá ningún recurso externo, evitando así dificultades en factores como escalabilidad y compatibilidad.

Mapas

La mayoría de los elementos que se analizaron tienen como objetivo principal poder adquirir una adaptabilidad que sea concisa con el tamaño de la pantalla y de acuerdo al dispositivo utilizado, esto cumplirá el objetivo de que sea responsivo. A pesar de esto, cuando se trata de mapas se puede observar que, mediante el uso de los mismos, este atributo a veces no resulta tan conveniente, dado que un mapa interactivo es casi como una segunda web embebida dentro de otra. Es importante aclarar que cuando se trata de localización y presentación de caminos, estos resultados dependen de las API con las cuales se están trabajando.

Es interesante entender qué opciones existen, para saber si alguna de ellas satisface las condiciones en las que se desarrolla este proyecto. Las mismas son:

- Óptima experiencia de usuario.
- Adaptabilidad del objeto a la web sin importar el dispositivo utilizado.
- Baja complejidad en lo posible.

Vinculación con aplicaciones nativas

Esta opción se descarta, ya que no cumple con ningunas de nuestras condiciones para el proyecto como escalabilidad de la solución y experiencia de usuario completamente independiente del dispositivo que utilice.

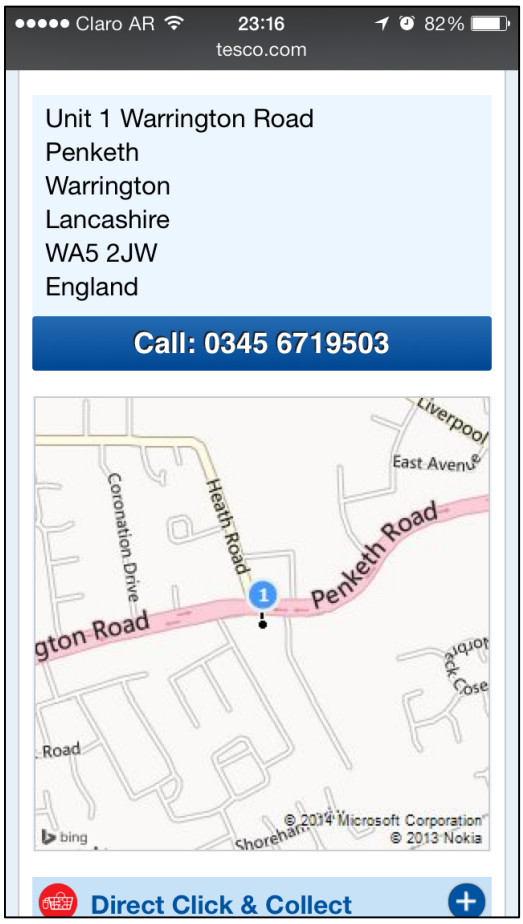


Figura 27: Aplicación web Tesco.com dirigiendo al usuario a la aplicación nativa de mapas (inicio).

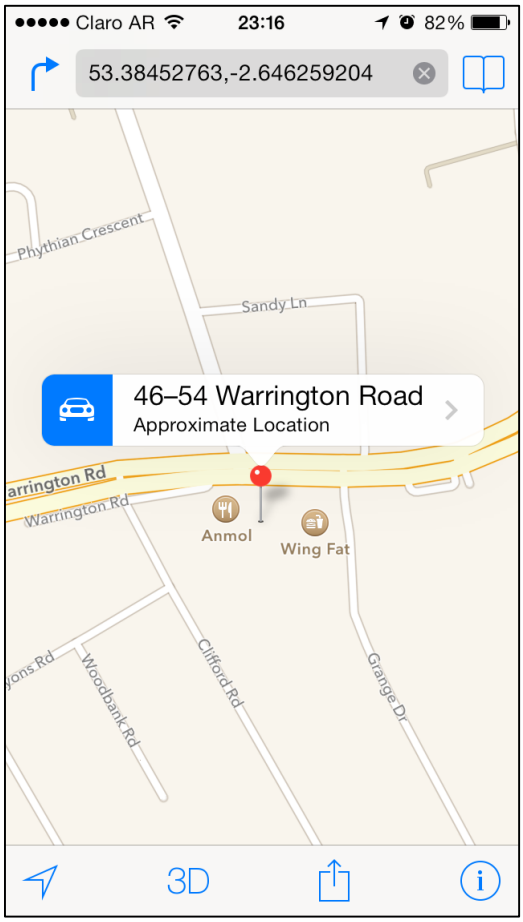


Figura 28: Aplicación web Tesco.com dirigiendo al usuario a la aplicación nativa de mapas (final).

Adaptabilidad al tamaño de la pantalla

Esta solución se basa en mostrar el contenido del mapa dentro de un contenedor fluido que ajuste su tamaño mientras el tamaño de la página cambia. Muchas veces esto presenta un problema con respecto a los dispositivos móviles ya que el objeto no se adapta correctamente y por este motivo el usuario encuentra dificultades en su uso.

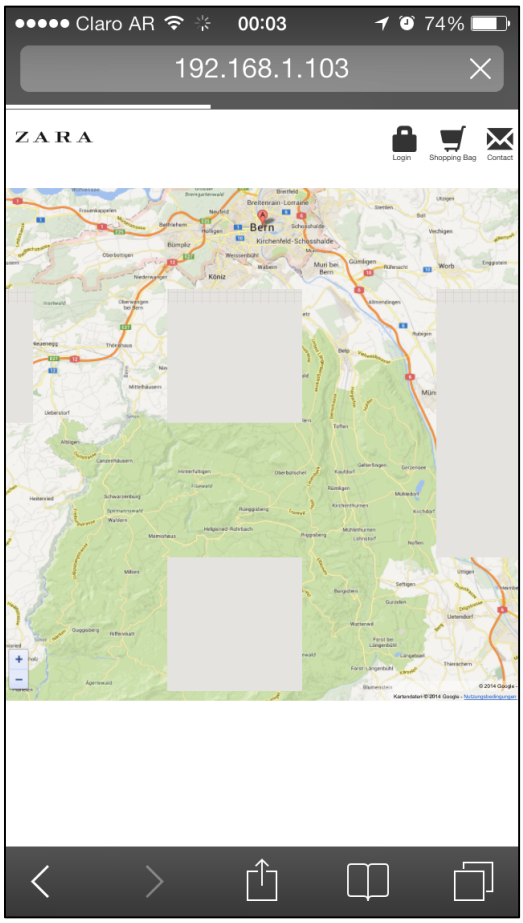


Figura 29: Problema nº1 al utilizar un objeto mapa embebido en una web.

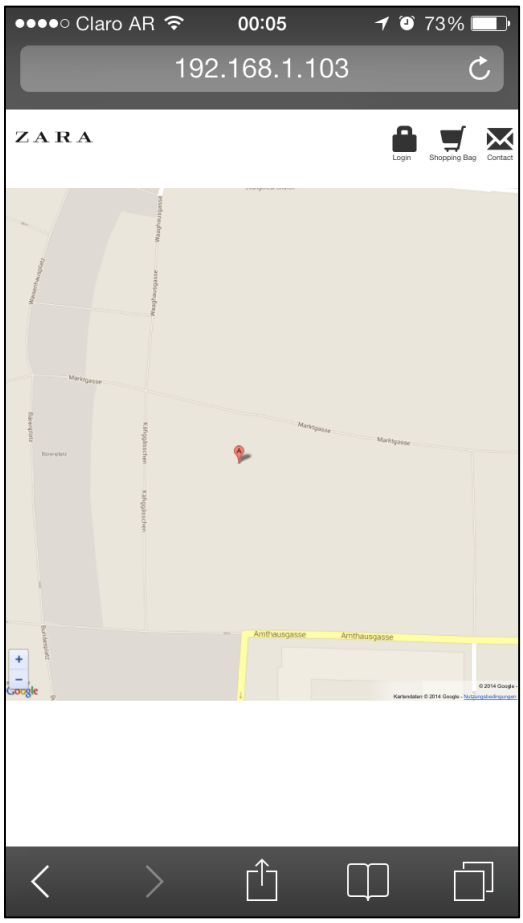


Figura 30: Problema nº2 al utilizar un objeto mapa embebido en una web.

Cada captura muestra un tipo de problema claro. La ilustración número 29 demuestra que pese a la posibilidad de poder mostrar en gran parte de la pantalla un mapa, al querer interactuar con el mismo genera dificultades de performance al requerir mucha capacidad de descarga de imágenes.

La segunda captura demuestra que pese a que se pueda interactuar dentro del mapa, al querer ampliar el mismo sigue siendo imposible leer aquellos textos que se encuentran como calles, zonas turísticas o algún otro punto denotado por el usuario. Esta interacción es a veces confundida, sobre todo si la web que embebe este objeto también tiene barras de desplazamiento verticales.

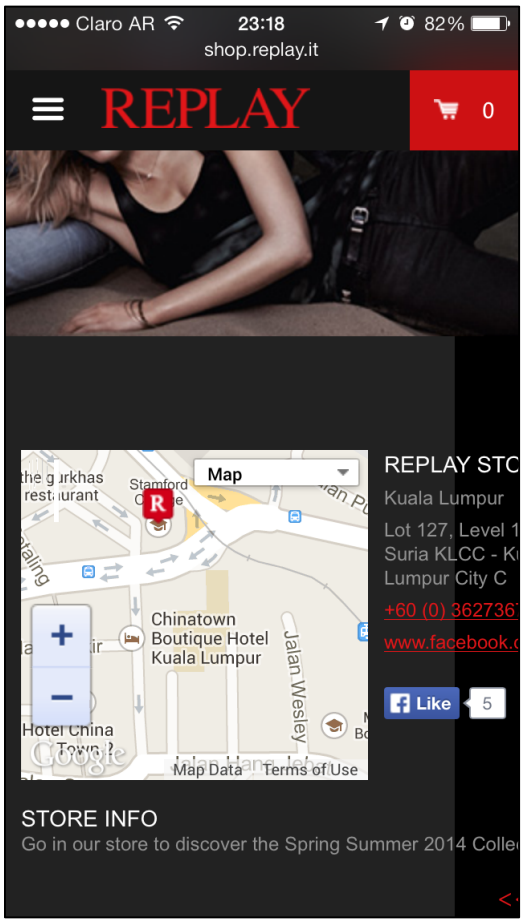


Figura 31: Mapa con poca visibilidad en web de Replay.

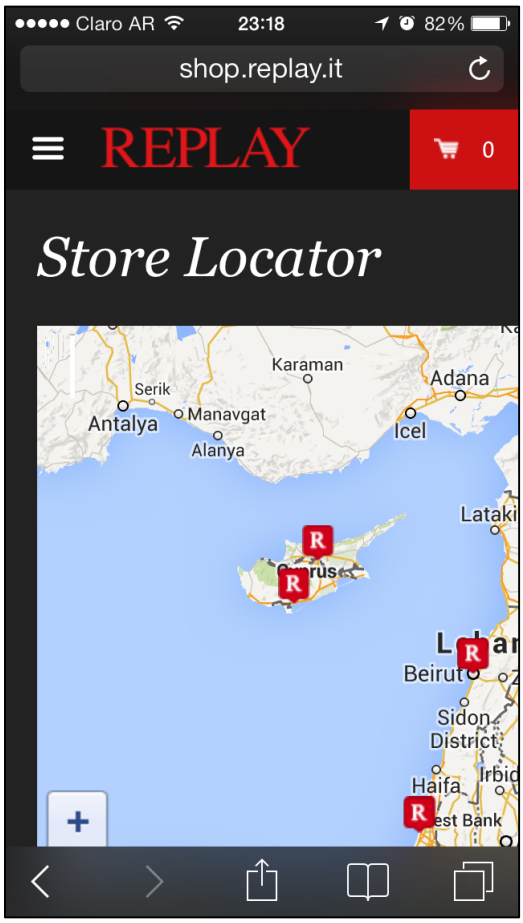


Figura 32: Mapa con zoom habilitado en la web de Replay.

Finalmente, en la transición entre la ilustración número 31 y 32 se demuestra la necesidad por parte del usuario de tener que realizar zoom en la página web para poder ampliar el mapa y luego interactuar con el zoom del mismo mapa para encontrar la ubicación.

Teniendo en cuenta lo anteriormente descripto, se pueden destacar las siguientes ventajas y desventajas:

Ventajas

- Todas las funciones habilitadas para la interacción con un mapa en una única página.
- Brindar la posibilidad al usuario de evaluar opciones a través de la interacción directa con el mapa.

Desventajas

- Complejidad de la aplicación.

- La experiencia de usuario puede verse afectada en usuarios novatos (al menos en dispositivos con tamaño de pantalla reducido).
- Lentitud de carga en la página.

Se puede concluir que, pese a que la adaptabilidad del mapa, la experiencia de usuario puede resultar ser totalmente pésima.

URL al mapa y mapas estáticos

Una solución simple sería la de brindar un vínculo al mapa, que hará que el usuario pueda utilizar aplicaciones como Google Maps o Bing en caso que la tuviese, o lo llevara a la web de ellos donde podría interactuar en pantalla completa. Existe también una herramienta llamada *Static maps* API que puede ser implementada para mejorar la experiencia de usuario mediante una imagen estática que sirva de vinculo para el mismo mapa.

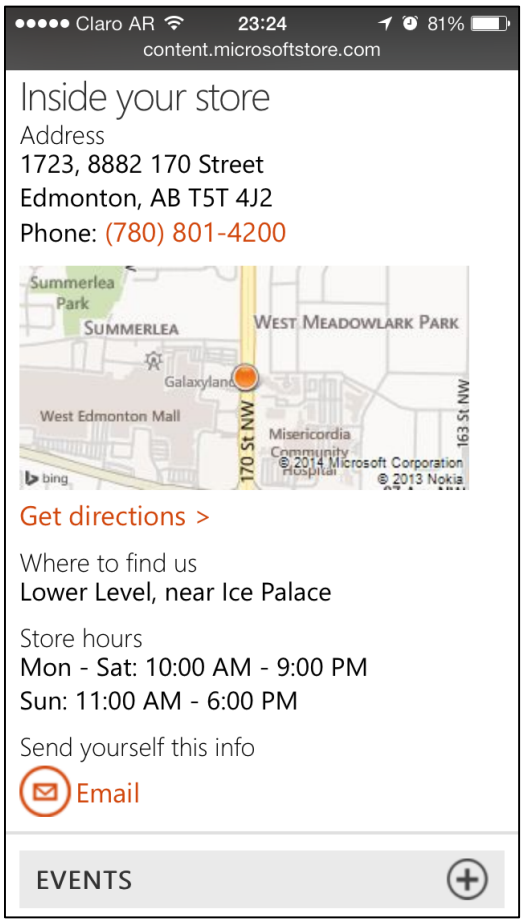


Figura 33: Mapas de Bing en modo estático.

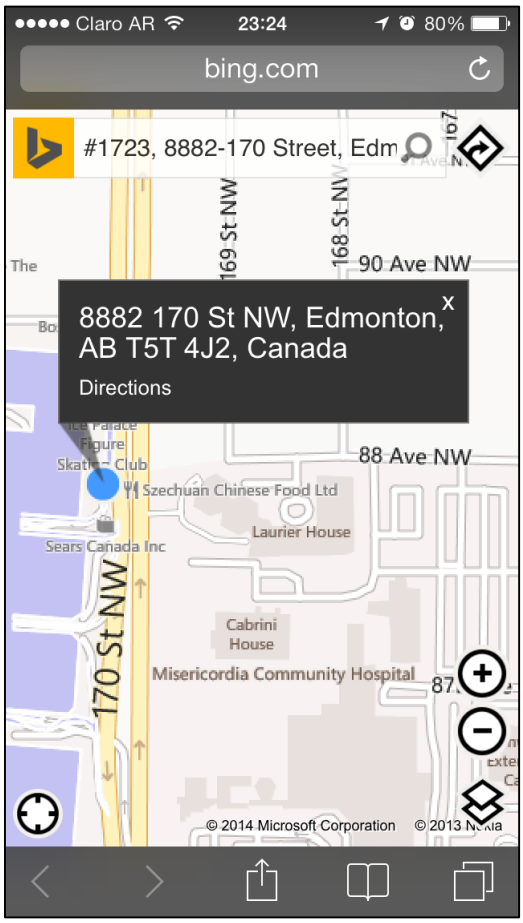


Figura 34: Mapas de Bing en modo dinámico.

También existe una extensión a dicha solución en la que se puede realizar una ampliación de pantalla completa del mapa dentro de la web donde se esté navegando. Con estas facilidades es posible determinar ciertas ventajas y desventajas:

Ventajas:

- Simplicidad en su navegación.
- Rapidez en cargar una imagen en vez de un mapa.

Desventajas:

- La utilización del mapa deberá hacerse en una página aparte. Usuarios novatos podrían ver esto como una complejidad de navegación.

Se puede concluir que esta opción invierte las ventajas y desventajas del punto 2. De todas maneras, es necesario encontrar un punto medio en el cual se puedan aprovechar ambas soluciones.

Detección de dispositivo o pantalla mediante Javascript

Consiste en detectar desde qué dispositivo se encuentra el usuario accediendo para poder determinar qué tipo de mapa brindarle. Se recomienda que si el dispositivo es de tipo móvil, se provea de una imagen estática como de la opción anterior brindando así una buena integración con un mapa completamente interactivo que pueda ser adaptado según los términos de *RWD*. En este caso es posible observar que las ventajas y desventajas se optimizan al máximo:

Ventajas:

- Simplicidad en su navegación.
- Al intercambiar el objeto del mapa con una imagen, la carga es rápida.

Desventajas:

- La utilización del mapa deberá hacerse en una página aparte en caso de ser un dispositivo pequeño destinado a la navegación simple.

Formularios

Introducción

Son parte de la interacción que se da con el usuario. Consisten en grupos de campo mostrados en forma de cajas por pantalla que se encargan de obtener los datos que el usuario le provea. Pueden ser en forma de caracteres alfanuméricos como también en objetos multimedia o archivos (según disponga).

Contemplar la posibilidad de adaptar formularios a RWD tiene un enfoque bastante acotado. Habiendo mencionado las distintas metodologías a aplicar en menús o imágenes descubrimos que es útil tomar lo bueno de cada una. En el caso de formularios, estos métodos no existen ya que hay una única forma de hacer que los mismos sean adaptables.

- Adaptabilidad de los campos al tamaño de la pantalla (elementos apilables).
- Campos con validaciones on-line visibles por los usuarios, manejo de errores.
- Campos de auto-ayuda.
- Proveer controles útiles al usuario para completar la información requerida de la manera más sencilla posible.

Para esto, analizaremos distintos tipos de formularios, contemplando sus diferencias para adaptarlas a RWD. Tendremos en cuenta lo que es un típico sistema de mails en el cual aparecen varios elementos contemplados en secciones anteriores como imágenes, menús, etc., pero aquí el tema principal se basará en lo relacionado a los formularios (inicio de sesión de usuario, validación de todos los campos, envío de e-mail, buscador de usuarios).

Adaptabilidad

La necesidad de adaptar un formulario a RWD es fácilmente descriptible mediante el primer formulario que se presentaría al querer consultar un sistema de e-mails: el inicio de sesión de usuario:

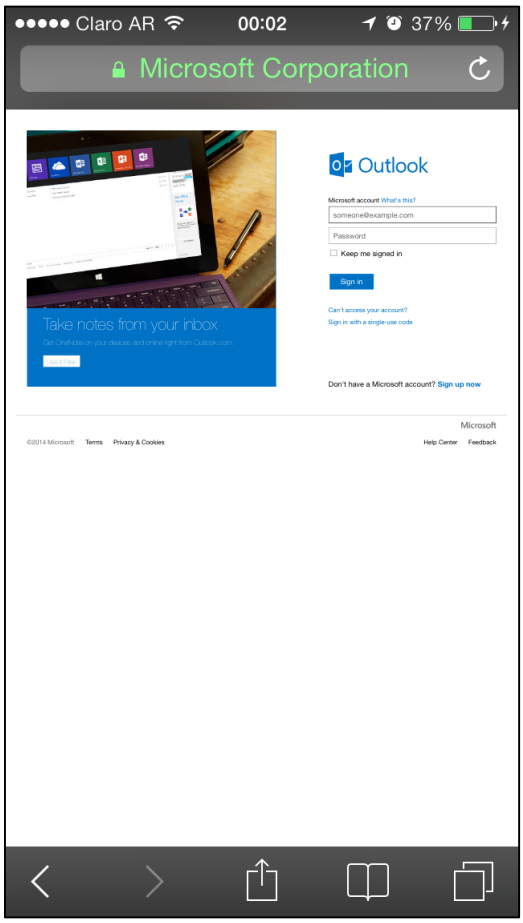


Figura 35: Formulario de inicio de sesión no ajustables automáticamente en Outlook.

A simple vista no parece tener un diseño apropiado para dispositivos móviles. Quizá sea fácil de navegar en un ordenador dado el tamaño del formulario. El texto del botón difícilmente podemos descifrarlo, de modo tal que un usuario que ingrese por vez primera no sabrá que acción tomar o de qué se tratará dicho portal.

Lo que intentaremos realizar es que sea adaptable, intuitivo y de uso sencillo, sin importar desde donde accedamos:

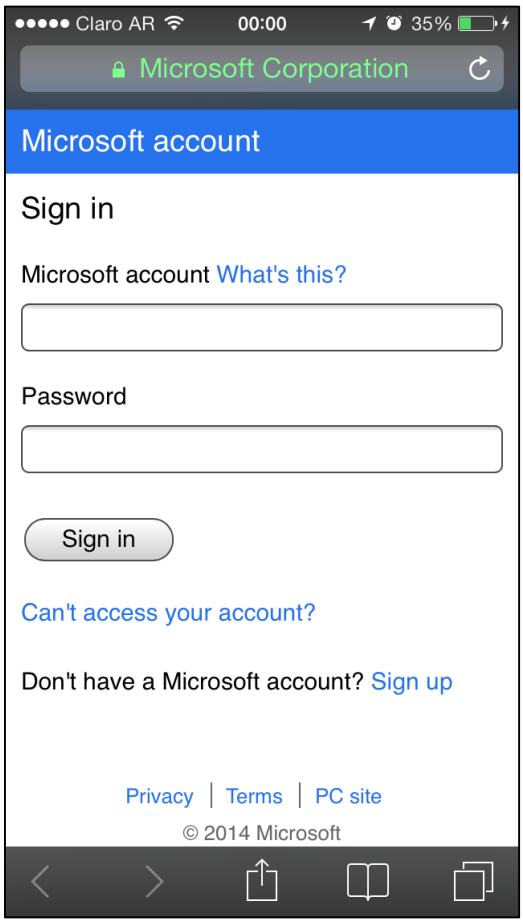


Figura 36: Formulario auto-ajustable en hotmail.com.

Es posible observar todos los problemas anteriormente mencionados resueltos en la ilustración 36. Teniendo en cuenta también que en pantallas de tamaño superior, la estructura del formulario puede ser de formato horizontal extendido, en dispositivos móviles deberán estar apilados para aparecer de forma correcta en la pantalla.

Validaciones y manejo de errores

Existen diversos puntos a tener en cuenta a la hora de validar información, detectar y notificar al usuario sobre los mismos, haciendo la experiencia más agradable y comprensible.

Todo formulario con campos requeridos deberá tener un control que verifique que hayan sido completados y contengan formato adecuado. Por ejemplo, al querer agregar destinatarios a un

mail, al querer enviar dicho mail, dicho campo deberá corroborar que contenga una casilla correcta, y no le falten caracteres esenciales de un mail "@".

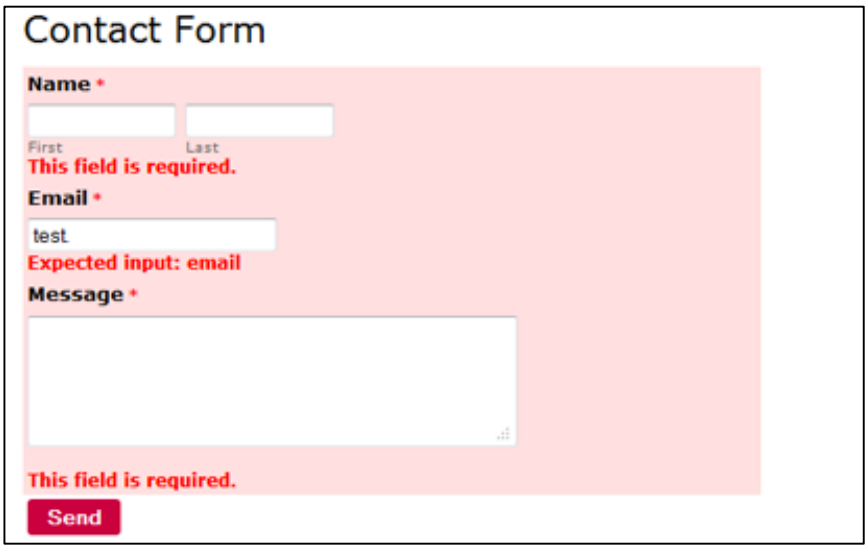


Figura 37: Ejemplo nº1 de formulario y validaciones on-line.

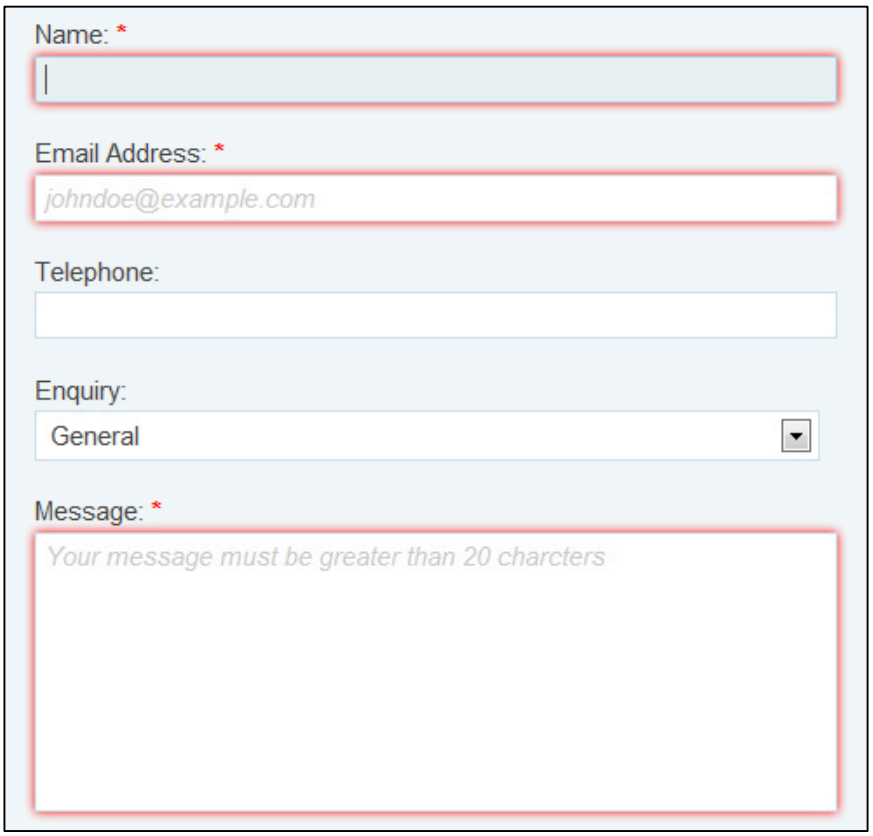


Figura 38: Ejemplo nº 2 de formulario y validaciones on-line.

Respuestas on-line

Esta funcionalidad, relativamente nueva en el mundo web, ha sido implementada muchas veces dada la facilidad y el resultado positivo de los usuarios frente a esta modalidad de intercambio de datos mediante formularios. Cuando un usuario envía sus datos, puede pasar dos cosas: la página carga completamente para mostrar los resultados de la consulta; el formulario procesa la petición y muestra los resultados sin realizar una carga completa de la página.

Esto ha sido de gran utilidad frente a los buscadores de cualquier tipo, en donde un usuario escribe su consulta y automáticamente obtiene resultados en pantalla mientras sigue escribiendo. Siguiendo el ejemplo de un sistema de e-mails, podemos observar esto mismo en el buscador de usuarios de una cuenta:

Figura 39: Búsquedas con escritura dinámica en Gmail.

Controles de usuario

Es importante entender que al tener una variedad de dispositivos que abarcar, los mismos serán distintos en varios aspectos, siendo uno de ellos el tipo de teclado que el usuario tiene disponible para ingresar datos. Algunos poseen botones especiales para facilitar la escritura de ciertos patrones como (@gmail, .com , //localhost). Existe un estudio en el cual se determina que estos teclados especiales no agregan un valor agregado real. Pese a ello, los usuarios deciden qué utilizar y qué no, por ello es necesario tenerlo en cuenta.

Usar elementos apropiados para los controles del usuario

Es ocurrente pensar que no solo basta con mostrar los elementos de manera prolija en los dispositivos mas pequeños; también hace falta identificar si aquellos elementos son adecuados para el tipo de interacción que el usuario necesita. Esto generalmente viene relacionado con los medios por los cuales el usuario interactúa con el dispositivo (Teclado fijo, teclado táctil, *trackball*). Puede que para aquellos usuarios que dispongan de un dispositivo con teclado fijo, el tener una lista larga de opciones a elegir le resulte mas difícil que a aquellos que naveguen con una pantalla táctil o un *trackball*. A pesar de que el impedimento provenga del hardware, esto no debería ser un impedimento para la optimización del mismo. La clave se encuentra en simplificar.

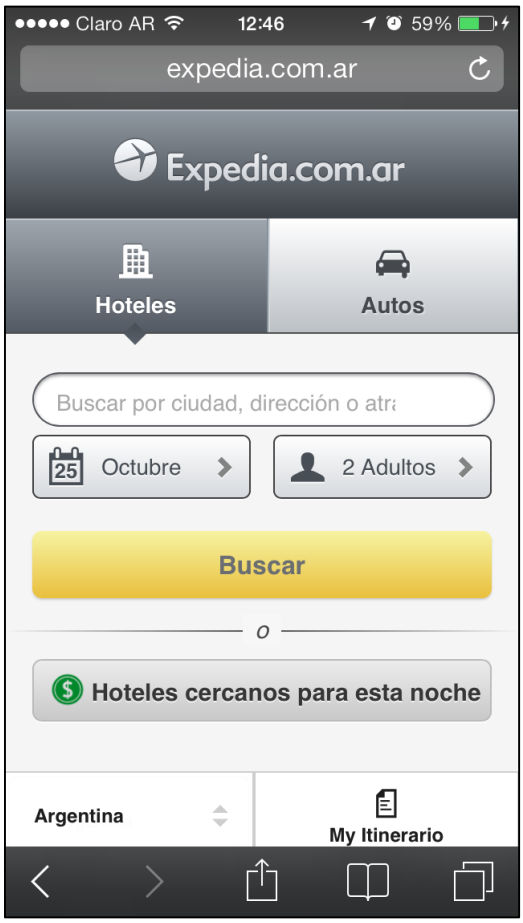


Figura 40: Aplicación web con controles adecuados.

En la página de Expedia un usuario puede utilizar sus servicios para buscar hoteles, viajes, renta de autos, buscar actividades planeadas, etc. Si un usuario utiliza este servicio desde una vista amplia, es decir desde una PC, verá que el formulario es amplio, con una diversidad de controles y vistas. La implementación móvil fue bien realizada considerando que el usuario deseará completar los campos lo más rápido posible. Se puede apreciar en la imagen cómo lograron adaptar los elementos ideales para cada tipo de datos; 2 solapas principales para elegir el motivo de la visita, una caja de búsqueda dinámica en la cual el usuario al ir escribiendo le aparecerán los resultados que espera encontrar, un calendario para elegir la fecha y un *drop-down* menú para elegir la cantidad de viajeros.

Figura 41: Skyscanner vista iPad sobre controles utilizados.

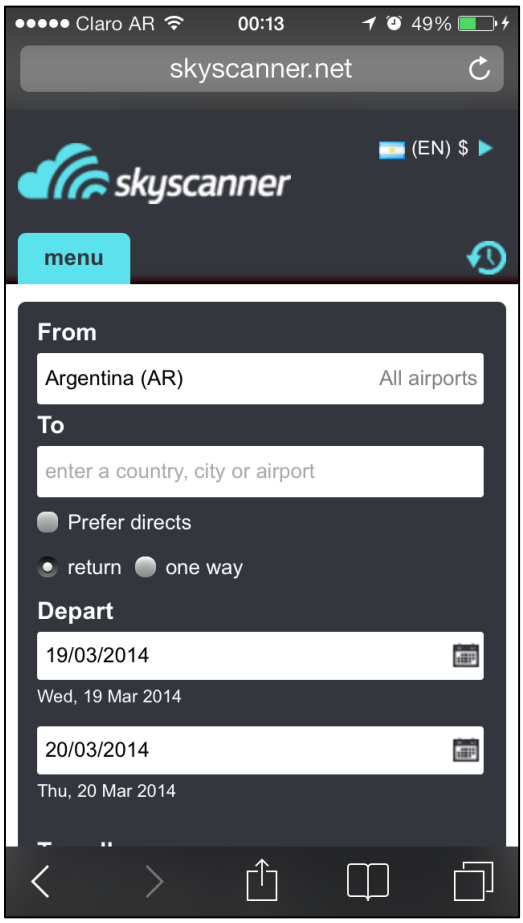


Figura 42: Skyscanner combinando varios controles en uno para simplificar la interfaz.

Si tomamos este último control desde la experiencia de usuario móvil, encontraremos una notable diferencia en cuanto al tiempo a utilizar, además de lo dificultoso. Por esto es que la web optó por remover varios controles, dividiendo sus opciones en distintos formularios, dándoles acceso mediante links distintos. Esto acota la cantidad de controles mostrados en pantalla, así como también facilita al usuario la elección de sus parámetros mostrando como resultado final una página más liviana para la carga en dispositivos móviles.

Elección del tipo de listas

Es importante la elección adecuada según el tipo de información a mostrar. Existen dos tipos de selección de opciones de un listado: Aquella en la cual se muestran todas las opciones posibles dentro de una lista de tamaño fijo, o aquella en la cual el campo a

completar es predictivo, es decir que a medida que el usuario ingresa caracteres, se van mostrando opciones que coincidan con ellas.

En la siguiente imagen se puede apreciar la diferencia que produce según el tipo de elementos a mostrar que no siempre es adecuado tener una lista fija. Para listar países el usuario ya entiende cuáles son sus opciones, por ser de conocimiento general. Distinto es el caso en la cual el grupo de opciones no tiene un orden determinado y las mismas no son familiares para el usuario. El mismo deberá analizar todas las opciones y optar por la mas adecuada para su uso. Cuanto mas larga sea esta ultima lista, mayor será el tiempo que necesitará el usuario para navegarla.

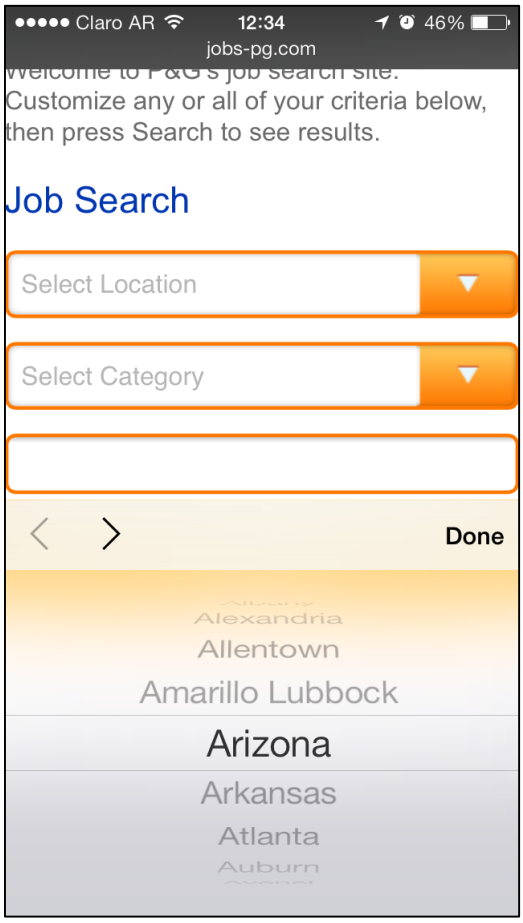


Figura 43: Ejemplo de menú dropdown.

Ayuda al usuario

Con el dinamismo que han adquirido últimamente las webs gracias a las tecnologías del momento, es posible adaptar una web para que parezca exactamente a una aplicación como si fuese nativa. Ya que es sumamente importante la interacción del usuario con la aplicación, es necesario incorporar niveles de ayuda hacia el mismo de modo tal que el aprendizaje sea acorde, y no falle en su intento.

Para poder lograr esto, existen varias formas de incorporar pequeñas (o grandes) ayudas paralelamente a la navegación del usuario; mediante pop-ups, introducciones, tutoriales.

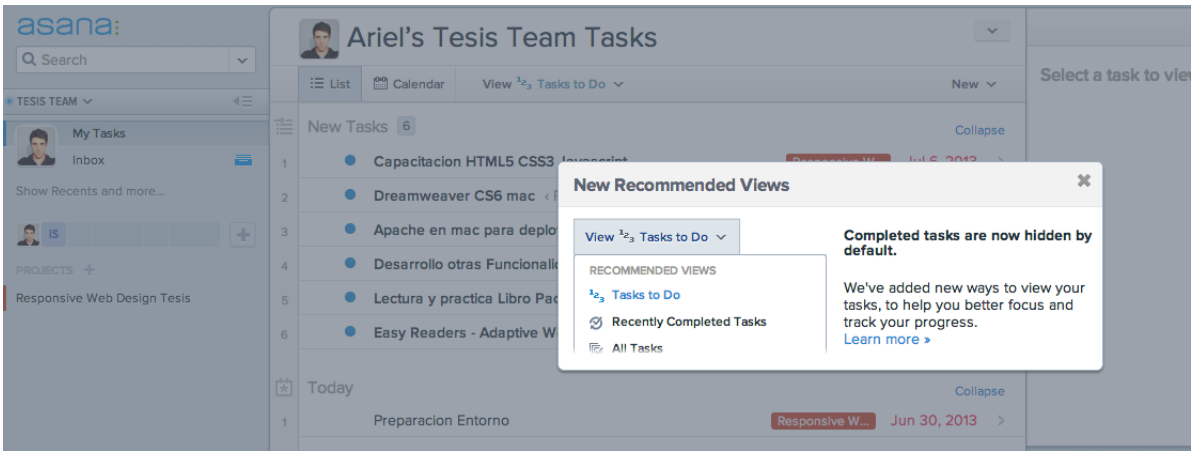


Figura 44: nuevas características de la aplicación en Asana se muestran en una ventana de introducción.

Cabe destacar que, como en todas las secciones anteriormente vistas, ésta también entra en el análisis para entender cómo debe comportarse según el tipo de dispositivo; la solución puede no ser exactamente la más conveniente al tratarse de un dispositivo móvil. Es conveniente, dado las dimensiones de la pantalla, que la ayuda se dé en una pantalla completamente aparte en caso de que la información a mostrar no sea pequeña.

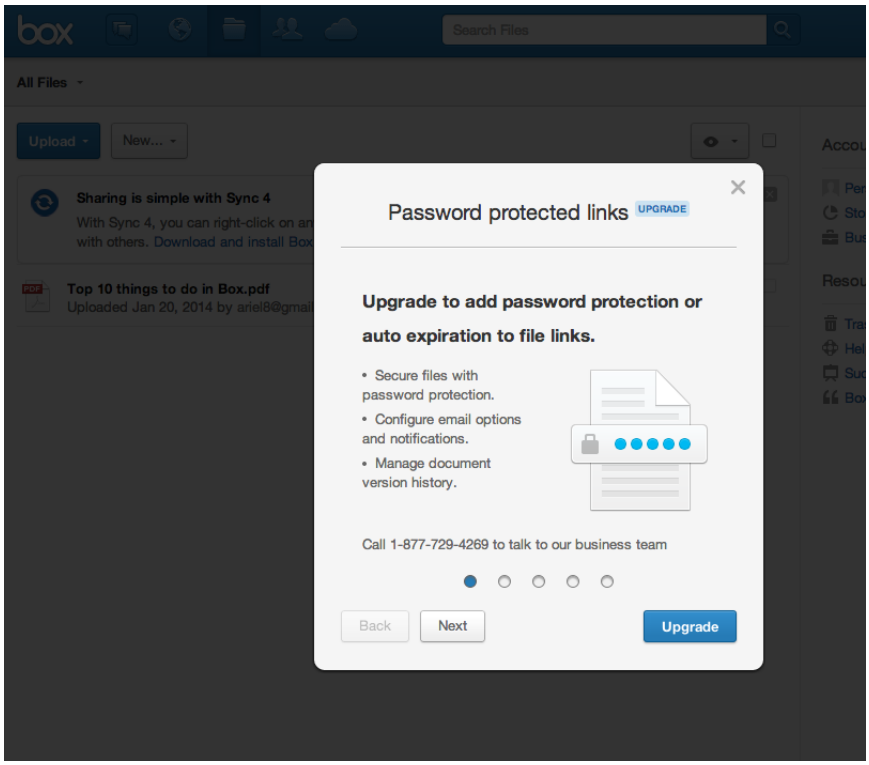


Figura 45: Ejemplo de Box con otra ventana de ayuda.

Video

Introducción

El video es uno de los elementos que más crecimiento tuvo en los últimos años con la aparición de proveedores tales como Youtube o Vimeo, o las mismas redes sociales como Facebook o Twitter.

Al igual que como ocurre con las otras secciones analizadas, la adaptación nunca estuvo entre los elementos a tener en cuenta a la hora de adaptarlas a otros medios. Videos en particular podría ser comparada con el modo en el que las imágenes interactúan en una página, las cuales presentan complicaciones del estilo:

- Adaptabilidad del tamaño del video según el ancho de la página.
- Calidad a mostrar debe ser conveniente para el tipo de dispositivo y/o pantalla.

No obstante, ésta presenta una particularidad; el elemento de video es un conjunto de funcionalidades (así como lo que ocurre en mapas) que están contenidas dentro de un cuadro en el cual se mostrara el video. De la misma manera sucede con la reproducción de audio y sin video, donde la manera en la que se realiza el *render* de los controles para manejar dicho contenido y la manera en la que estos contenidos son administrados presenta cierta dificultad. La importancia aquí está en poder elaborar una solución que sea capaz de mostrar el total de las funcionalidades, así como el contenido del video, de forma correcta, eficiente y con adaptabilidad frente a cualquier dispositivo.

En cuanto a la adaptabilidad del objeto en su totalidad no existen problemas mayores; al estar dentro de un elemento cuadro es cuestión de utilizar las mismas metodologías provistas en la sección de Imágenes y aplicarlas a ésta, de modo tal de desarrollar la adaptabilidad frente a distintos tamaños de pantalla / ventana.

Los retos que se encuentran en estas otras dificultades se pueden presentar como:

- Videos de proveedores externos (Vimeo, Youtube).
- Aspectos de video distintos (16:9, 4:3).
- Navegadores que aún no estandarizan los formatos de videos soportados.

- Funcionalidades habilitadas/deshabilitadas según el tamaño de pantalla.

No es menor el detalle de que hoy en día existen diversos navegadores que son utilizados por los usuarios, y que aún no contienen todos los mismos estándares, es decir, no soportan los mismos formatos, extensiones, y codificaciones de video.

BROWSER/DEVICE	VIDEO FORMATS	AUDIO FORMATS	MULTIPLE SOURCES
Chrome	MP4, WebM	AAC, MP3, Vorbis	✓
Firefox	MP4, WebM	AAC, MP3, Vorbis	✓
Internet Explorer	MP4	AAC, MP3	✓
Android	MP4	AAC, MP3	✓
iOS	MP4	AAC, MP3	✓
Safari	MP4	AAC, MP3	✓

Figura 46: Tabla de formatos soportados por cada navegador.

BROWSER/DEVICE	MARKET SHARE	HTML5 VIDEO	FLASH VIDEO
Chrome	33%	✓	✓
Firefox	14%	✓	✓
Internet Explorer 9+	12%	✓	✓
Android	11%	✓	✗
iOS	9%	✓	✗
Safari	4%	✓	✓
Other (desktop)	8%	✗	✓
Other (mobile)	10%	✗	✗

Figura 47: Soporte de distintas tecnologías de video por navegador.

Volviendo al objeto del estudio, la idea sería apuntar a lograr la flexibilidad de una aplicación como Netflix, claro ejemplo en el que se implementan diversas estrategias para proveer compatibilidad con la mayoría de los navegadores existentes.

Es importante destacar que al igual que en imágenes, existen dos soluciones para poder presentar videos en una página web. Cada una de ellas presenta soluciones especificas así como también ciertas ventajas y desventajas. Sin embargo, estas son excluyentes y no pueden ser utilizadas al mismo tiempo para la transmisión de un mismo video, aunque si se pueden intercalar en una misma web para distintos videos en caso de ser necesario y admisible.

Self-hosted videos

Este método requiere que el desarrollador exponga la lógica, los recursos y la forma en la que se mostrará la interfaz del elemento video. Desde HTML5 existe la posibilidad de agregar estos objetos sin necesidad de agregar código Javascript.

Tenemos un mayor control del tipo de reproductor que deseamos utilizar, es decir, podemos hacer modificaciones personales según la complejidad o simplicidad que deseemos. Como contrapartida, tendremos que decidir qué *codecs* utilizar según el navegador, el aspecto que tendrá el video en cuanto al ancho y alto, etc.

Por otra parte, como bien se ha dicho, la lógica expuesta requerirá de la elección del tipo de recurso a mostrar según el tipo de explorador que el usuario este utilizando, así como el tipo de calidad.

No es objeto de este documento demostrar la retro-compatibilidad con navegadores obsoletos, pero cabe mencionar en este tema que es posible hacer una recuperación del video, frente a un fallo de la herramienta principal, hacia tecnologías no tan usadas como Adobe Flash o Microsoft Silverlight. Esto significa que aquellos usuarios con dispositivos obsoletos o con plataformas que no respetan los estándares, puedan gozar de la misma interfaz sin siquiera detectar el cambio. Por supuesto que esta última característica requerirá de una lógica adicional.

Las modificaciones personales del controlador pueden llegar a tal nivel de poder mostrar un video sin la necesidad de mostrar controles, es decir que reproduzca

directamente cuando el usuario carga la página. Dicho video puede mostrarse como fondo en las webs, lo cual es una técnica utilizada últimamente como una forma de enriquecer la experiencia del usuario en la navegación.

Videos embebidos de terceros

Esta opción es generalmente la mas utilizada. Consiste en utilizar los recursos ya brindados por webs dedicadas a la publicación de contenido en formato de video como Vimeo o Youtube, mediante la referencia directa a ellas simplemente por medio de una URL.

Generalmente bastaría con conocer la ruta completa del archivo de video que se desee referenciar y utilizar la metodología expuesta anteriormente. No sucede así con los grandes proveedores, los cuales no exponen dichas rutas ni nombres de archivos fijamente. Como alternativa, proveen de una API y el código HTML para poder utilizarlo en cualquier web.

En dicho código se especifican las posibles características a modificar tales como el aspecto (16:9), el ancho y alto específico así como también los botones a mostrar, etc. Lo que no se puede modificar en el controlador son aquellos referentes al proveedor en sí.

Figura 48: Video formato wide con menú inferior.

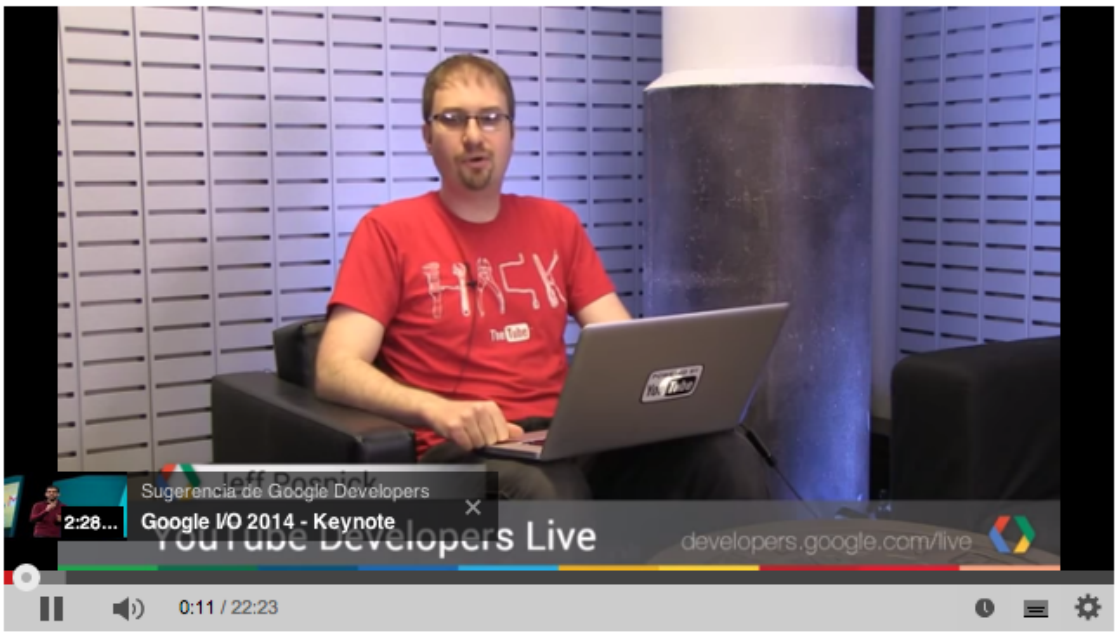


Figura 49: Video formato cuadrado sin menú inferior.

Pese a que esta metodología está controlada en definitiva por el proveedor, existe la forma de soportar también aquellos usuarios excepcionales que no se encuentren aun compatibles con HTML5. La solución está basada en proveer el video en formato Flash o JavaScript.

Conclusión

Será una de las tareas más complicadas poder resolver la muestra de videos de forma responsiva en cualquier dispositivo, sin tener que recurrir a herramientas externas para proveernos de soluciones, es decir, los navegadores de hoy en día tienen este punto resuelto pero es por ello que la incompatibilidad entre distintos navegadores dificulta la escalabilidad de los objetos dispuestos en una web para el desarrollador o diseñador.

Conclusiones del trabajo

Pese a la inmensa cantidad de información existente hoy en día, y el acceso simple de que se dispone, no se encontrará una totalidad de proyectos en los cuales se tomen recaudos importantes respecto a la interacción del usuario frente a ellos. Podemos constatar que el tema de *RWD* es uno de los aspectos más importantes que hoy en día se suma a los nuevos proyectos, pero que falta concientización en los ya existentes para la adaptación de los conceptos, que no serán una pérdida de tiempo sino más bien un valor agregado fundamental.

Por otro lado, se denota que las tecnologías de hoy en día y las que irán apareciendo indirectamente, lo vendrán incorporando en la labor de los desarrolladores y diseñadores que aun tendrán una amplia gama de opciones para brindar soluciones, pero que quedará en ellos tener el interés y el conocimiento para poder elegir la más eficiente.

Bibliografía

Alexander, S., Choosing a Responsive Image Solution, http://www.smashingmagazine.com/2013/07/08/choosing-a-responsive-image-solution/, 2013.

Andersen, A., Addressing The Responsive Images Performance Problem: A Case Study, http://www.smashingmagazine.com/2013/09/16/responsive-images-performance-problem-case-study/, 2013

AWWWARDS, Infographic: Responsive Images Problems and Solutions, http://www.awwwards.com/gallery/9078/infographic-responsive-images-problems-and-solutions, 2014.

Barrett, M., Build a Neat HTML5 Powered Contact Form, http://code.tutsplus.com/tutorials/build-a-neat-html5-powered-contact-form--net-20426, 2011.

Baxter, K., HTML5 Video Playback UI, http://techblog.netflix.com/2013/10/html5-video-playback-ui.html, 2013.

Birch, N., Vertical Navigation, Implementation of Side Menu in Mobile Apps, http://designmodo.com/vertical-side-menu-mobile-apps/, 2013.

Bradley, S., 3 Type of Solutions to Work With Responsive Images, http://www.vanseodesign.com/web-design/responsive-images/, 2012.

Brightcove, Responsive Sizing for Video Players, http://docs.brightcove.com/en/video-cloud/smart-player-api/samples/responsive-sizing.html, 2014.

Bushell, D., Responsive Calendar, http://dbushell.com/demos/calendar/v1_03-01-12.html, 2012.

Bushell, D., The Raster Image Paradox, http://dbushell.com/2013/06/03/the-raster-image-paradox/, 2013.

Caldwell, A., 2013 Web Design Trends, http://brolik.com/blog/2013-web-design-trends/, 2013.

Coron, T., How to create a Slide-Out Navigation Panel, http://www.raywenderlich.com/32054/how-to-create-a-slide-out-navigation-like-facebook-and-path, 2013.

Coyier, C., Responsive Data Tables, CSS-TRICKS, http://css-tricks.com/responsive-data-tables/, 2011.

Firdaus, T., Responsive Web Design by Example, PACKT Publishing, pág. 67-74, 2013.

Forde, D., HTML5 Facebook Style Sliding Menu Using Twitter Bootstrap Collapse, http://tech.xtremelabs.com/html5-facebook-style-sliding-menu-using-twitter-bootstrap-collapse/, 2013.

Frain, B., Responsive Web Design with HTML5 and CSS3, pág. 35-97, pág. 237-269, PACKT Publishing, 2013.

Galitz, W., The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 3ra Ed, pág. 545, Wiley, 2007.

Gatlin, R., Responsive Image Solutions, http://www.pragermicrosystems.com/blog/web-development/responsive-image-solutions/, 2013

Gergeo, N., Responsive Tables, http://gergeo.se/RWD-Table-Patterns/#demo, 2014.

Grigsby, J., CSS Media Query for Mobile is Fool's Gold, http://blog.cloudfour.com/css-media-query-for-mobile-is-fools-gold/, 2010.

Gustafson, A., Adaptive Web Design, Crafting Rich Experiences with Progressive Enhancement, pág. 12, Easy Readers, 2011.

Hannemann, Anselm, Demos, http://responsiveimages.org/demos/, 2014.

Humphreys, J., Say Goodbye to Painful Image Load on Small Devices. Say Hello to Interchange, http://zurb.com/article/1211/say-goodbye-to-painful-image-loads-on-sma, 2013.

Jenkins, S., Create fullscreen HTML5 responsive video backgrounds, http://www.webdesignermag.co.uk/tutorials/create-fullscreen-html5-responsive-video-backgrounds/ ,2014.

Kadlec, T., Implementing Responsive Design: Building sites for an anywhere, everywhere web, New Riders, 2012.

Kadlec, T., Why We Need Responsive Images, http://timkadlec.com/2013/06/why-we-need-responsive-images/, 2013.

Khan, A., Implementing Container Containtment, http://code.tutsplus.com/tutorials/implementing-container-containment-sliding-menu-controller--mobile-14562, 2013.

Lawson, B., Why HTML5 urgently needs and HTML adaptive images mechanism, http://www.brucelawson.co.uk/2012/html5-urgently-needs-adaptive-images-mechanism/, 2012.

Martin, D., Fat Fingers: A Look Into Responsive Navigations Trends, http://designwoop.com/2013/07/responsive-navigation-trends/, 2013.

Marcotte, E., Responsive Web Design, A Book Apart, 2011.

McCollin, R., Responsive iFrames, http://rachelmccollin.co.uk/blog/, 2013.

Microsoft, Menus, http://msdn.microsoft.com/en-us/library/windows/desktop/dn742392.aspx, 2014.

Miller, D., Adaptive images: solving the responsive image problem, http://www.webdesignerdepot.com/2012/07/adaptive-images-solving-the-responsive-image-problem/, 2012.

Park, A., Watson, M., HTML5 Video at Netflix, http://techblog.netflix.com/2013/04/html5-video-at-netflix.html, 2013.

PC.net, Drop Down Menu, http://pc.net/glossary/definition/dropdownmenu, 2010.

Perez, Y., FocalPoint, http://stashy.azurewebsites.net/focalpoint, 2013.

Pogson, U., Complete Breakdown of Responsive Videos, https://ulrich.pogson.ch/complete-responsive-videos-breakdown, 2013.

Purcell, K., The State of Online Video, http://www.pewinternet.org/files/old-media/Files/Reports/2010/PIP-The-State-of-Online-Video.pdf, 2010.

Ramos, M., Responsive Images: Simple Solutions, http://lucentwebdesign.co.uk/index.php/blog/entry/responsive_images_the_simplest_solutions_are_the_best_solutions, 2012.

Rocheleau, J., Popular Design for Responsive Navigation, http://www.vandelaydesign.com/responsive-navigation/, 2013.

Schmitt, C., The problema with adaptive images, http://www.creativebloq.com/css3/problem-adaptive-images-5126295, 2012

Souders, S., I <3 image bytes, http://www.stevesouders.com/blog/2013/04/26/i/, 2013.

Tan, C., Mobile Form Design Strategies, http://www.uxbooth.com/articles/mobile-form-design-strategies/, 2011.

Spurlock, J. Bootstrap, Responsive Web Development, pág. 41-72, pág. 78-79, pág. 91-92, O'Really, 2013.

Suh, J., Responsive YouTube, Vimeo, Embed, and HTML5 video with CSS, http://jonsuh.com/blog/responsive-youtube-vimeo-embed-and-html5-video-with-css/, 2012.

SundaySky, 2012 State of Online Video, http://www.slideshare.net/SundaySky/2012-state-of-online-video, 2012.

Tobon, B., iPhone sliding menu (Facebook style), http://blog.sallarp.com/iphone-sliding-menu/, 2008.

Tympanus.net, Responsive Full Width Tabs, http://tympanus.net/Blueprints/FullWidthTabs/, 2013.

Van Damme, B., Focal Point: Intelligent Cropping of Responsive Images, http://www.bram.us/2012/12/05/focal-point-intelligent-cropping-of-responsive-images/, 2012.

Vanderburg, S., Responsive Design image optimization with JavaScript in Media Queries, http://skipvanderburg.tumblr.com/post/42756483547/responsive-design-image-optimization-with-javascript-in, 2012.

Wijering, J., The State of HTML5 Video, http://www.jwplayer.com/html5/, 2014.

Wilcox, M., Adaptive Images, http://adaptive-images.com/, 2013.

Wroblewski, Mobile First, Numero 6, pág. 25, 2013.

Yarmosh, K., New iOS Design Pattern: Slide-Out Navigation, http://kenyarmosh.com/ios-pattern-slide-out-navigation/, 2012.