Gerencia de Desarrollo de Software

Guía de Mejores Prácticas para el Desarrollo Web

Febrero 26, 2015

Histórico de Versiones

| **Fecha** | **Versión** | **Descripción** | **Autor** |
| --- | --- | --- | --- |
| 26/02/2015 | 1.0 | Creación del documento | Francisco Manuel Carrasco L.  Isidro López Sosa  Martín Martínez Soto  Sergio Velázquez Sánchez |

Tabla de Contenido

[1. Introducción 5](#_Toc412797113)

[2. Objetivo 5](#_Toc412797114)

[3. Diseño Web 6](#_Toc412797115)

[3.1. Maquetación Web con Bootstrap 6](#_Toc412797116)

[4. Desempeño Web 8](#_Toc412797117)

[4.1. Generar archivos JavaScript y CSS por separado 8](#_Toc412797118)

[4.2. Minimizar las Hojas de estilo y los scripts. 8](#_Toc412797119)

[4.3. Combinar Scripts (JavaScript) 8](#_Toc412797120)

[4.4. Combinar las hojas de estilos (CSS) 8](#_Toc412797121)

[4.5. Hojas de estilos 9](#_Toc412797122)

[4.6. Documentar CSS 9](#_Toc412797123)

[4.7. Evitar el uso de Inline de estilos o scripts 10](#_Toc412797124)

[4.8. Colocar las hojas de estilo al inicio de las páginas (Header) 10](#_Toc412797125)

[4.9. Colocar los Scripts al final de la página (Footer) 11](#_Toc412797126)

[4.10. Optimizar las imágenes 11](#_Toc412797127)

[4.11. Alternativa de carga de imágenes 12](#_Toc412797128)

[4.12. Despliegue de animaciones 12](#_Toc412797129)

[5. Integración de componentes JSF 13](#_Toc412797130)

[5.1. Usar Controles JSF y evitar componentes de  PrimeFaces a medida de lo posible. 13](#_Toc412797131)

[5.2. Cambiar la propiedad “prependId” a false. 13](#_Toc412797132)

[5.3. Evitar FORMs anidados. 13](#_Toc412797133)

[5.4. Agregar a todos los controles un identificador 14](#_Toc412797134)

[5.5. Agregar la propiedad layout a block si utilizas h:panelGroup. 14](#_Toc412797135)

[5.6. Nombrar los componentes de forma coherente 14](#_Toc412797136)

[5.7. Reutilizar componentes. 14](#_Toc412797137)

[5.8. Utilizar h:panelGroup para realizar render de secciones 14](#_Toc412797138)

[5.9. Utilizar h:message 14](#_Toc412797139)

[5.10. No usar Scriptlets 14](#_Toc412797140)

[6. Diseño Responsivo 15](#_Toc412797141)

[6.1. Diseñar por porcentajes, no por pixeles. 15](#_Toc412797142)

[6.2. Diseñar para pantallas, no para dispositivos. 15](#_Toc412797143)

[6.3. Diseñar para tocar, no para clicks. 16](#_Toc412797144)

[6.4. Diseñar para pantallas pequeñas primero. 16](#_Toc412797145)

[6.5. Los usuarios móviles merecen la misma calidad y experiencia en la navegación. 16](#_Toc412797146)

[6.6. Diseña tu sitio pensando en “responsivo” 16](#_Toc412797147)

[6.7. Utilizar ems para la tipografía 17](#_Toc412797148)

[6.8. Las imágenes en móvil y grandes resoluciones 17](#_Toc412797149)

[6.9. Imágenes en dispositivos Retina Display 18](#_Toc412797150)

[6.10. Compatibilidad de SVG en navegadores antiguos 18](#_Toc412797151)

[7. Referencias 18](#_Toc412797152)

[8. Firmas de Elaboración y Revisión 20](#_Toc412797153)

**Guía de Mejores Prácticas para el Desarrollo Web**

# Introducción

Si bien es cierto que existen estándares que regulan el  desarrollo Web, también es cierto que cada equipo de desarrollo e incluso cada integrante del equipo, tiene su propia forma y estilo de desarrollar, sin embargo a nivel organizacional se deben generar los lineamientos necesarios que ayuden a marcar una línea de desarrollo con la finalidad de generar productos consistentes y de calidad.

En este documento se enuncian algunas de las mejores prácticas de desarrollo web entre las cuales se mencionan el diseño web, desempeño web, integración de componentes, y las mejores prácticas para el desarrollo de sitios responsivos.

# Objetivo

La presente guía tiene como objetivo que en las diferentes áreas de la Gerencia de Desarrollo de Software Web, se cuente con el conocimiento de las mejores prácticas de desarrollo para su implementación en los diferentes proyectos que se tienen.

# Diseño Web

### Maquetación Web con Bootstrap

**Descripción:** Esta sección está dedicada a las mejores prácticas de integración del framework  bootstrap con la finalidad de crear interfaces que se adapten a los distintos navegadores y dispositivos. Además de proveer de lineamientos para la estandarización y el uso correcto del mismo.

Bootstrap es una herramienta que facilitara el diseño responsivo, la maquetación y distribución de componentes.

Características principales de Bootstrap:

1. Uso de diseño web responsivo.
2. Uso de estándares HTML5 y CSS3
3. Sistema de retícula con 12 columnas
4. Integración fácil con librerías como jQuery

**Objetivo:** Delimitar los lineamientos para  la distribución de componentes

* En ocasiones es necesario encerrar cierto contenido de formularios dentro de un contenedor (div, span, etc.). En estos casos se deberá considerar el uso de las clases del tipo panel, es importante mencionar no incluir un panel dentro de otro panel.

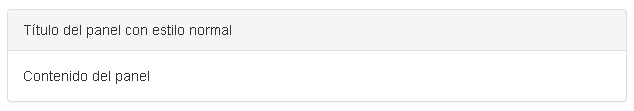
<div class="panel panel-default">

<div class="panel-heading">Título del panel con estilo normal</div>

<div class="panel-body">

Contenido del panel

</div>  
</div>

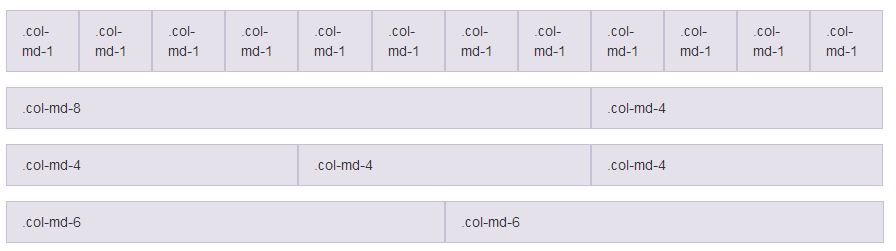


* Para el contenedor de tipo panel es recomendable utilizar las siguientes clases de retícula basada en 12 columnas.
* Utilizar .row para crear grupos horizontales de columnas.
* Las clases .row deben ser colocadas dentro de un .container
* Existen clases predefinidas como .row y .col-\*-\* las cuales están disponibles para hacer rápidamente diseños de cuadrícula.
* Las columnas en la cuadrícula se crean especificando el número de 12 columnas disponibles que desea abarcar. Por ejemplo, tres columnas iguales usarían tres .col-md-4

<div class="container">

  <div class="row">  
    <div class="col-md-4">...</div>  
    <div class="col-md-4">...</div>  
    <div class="col-md-4">...</div>  
  </div>

</div>



* El contenido debe ser colocado dentro de columnas (.col-\*-\*)  y estas deben de estar alojadas dentro de filas (.row).
* No puede existir una columna (.col) sin la existencia de una fila (.row), dentro de las columnas (.col-\*-\*) es donde se encuentra la información, como  elementos <input>, <textarea> y <select>, componentes de primefaces, texto imágenes etc.

***Nota:***

Para un mejor resultado en la maquetación se recomienda no utilizar tablas para crear formularios o contenido.

# Desempeño Web

**Descripción:** El rendimiento de las páginas web es una característica más en el desarrollo de aplicaciones, presentar el contenido lo más rápido posible, facilitar la interacción y procesamiento debe ser una meta al desarrollar nuestras aplicaciones mejorando así la experiencia de los usuarios.



**Objetivo:** Dar a conocer las mejores prácticas para mejorar el rendimiento de una página web, utilizando la tecnología de Java Server Faces (JSF).

### Generar archivos JavaScript y CSS por separado

Esto se debe a que los archivos CSS y JavaScript que se cargan  por primera vez se almacenan en el  caché del navegador y posteriormente las cargas de contenido se realizarán más rápido.

### Minimizar las Hojas de estilo y los scripts.

Apoyándose de herramientas que faciliten la compresión de código (Ej. CSS Compressor & Minifier), Procesar los archivos CSS y JS para mejorar los tiempos de carga en ambientes productivos.

### Combinar Scripts (JavaScript)

Múltiples scripts deben ser combinados en un solo archivo, lo ideal sería que no existiera más de un script que contenga toda la funcionalidad requerida para el aplicativo.

### Combinar las hojas de estilos (CSS)

Múltiples hojas de estilo deberán combinarse en una sola hoja y lo ideal es que exista solo una hoja de estilos para todo el aplicativo.

### Hojas de estilos

Cuidar el formato CSS con selectores en una línea y cada propiedad en su propia línea y las declaraciones con sangría.

.post-list li a{

color:#A8A8A8;

}

.post-list li a:hover{

color:#000;

text-decoration:none;

}

.post-list li .author a,

.post-list li .author a:hover{

color:#F30;

text-transform:uppercase;

}

### Documentar CSS

Documentar las estructuras complejas del código CSS, para su comprensión y mantenimiento por personas que no generaron el código.

Ejemplo código no documentado:

#header {}  
    #header h1{  
        color: red;  
    }

Está claro que sin saber que es #header, puede ser difícil mantener este código CSS.

/\*\*  
\* @style       Standard Layout  
\* @media       screen  
\* @version     1.0  
\* @author      Franky  
\* @copyright   Franky’s pwn comp-a-ni  
\* @licensor    da customa  
\* @layout      in pixels:  
\*              |            788            |  
\*              | 10  |      768       | 10 |  
\*              | 10  | 27 |    741    | 10 |  
\*/  
[...]  
/\*\*  
\* @section reset  
\* @see     …  
\* @see     http://meyerweb.com/eric/thoughts/2007/05/01/reset-reloaded/  
\*/  
[...]  
/\*\*  
\* @section layout  
\* @todo    margins an parent anpassen  
\*/

### Evitar el uso de Inline de estilos o scripts

Por defecto los archivos CSS y JavaScript se almacenan en el caché del navegador.

Si se utilizan scripts u hojas de estilo inline, se deberán descargar cada vez que se genere la petición no haciendo uso del caché del navegador lo cual reduce la velocidad de carga de la página.

### Colocar las hojas de estilo al inicio de las páginas (Header)

* Las hojas de estilo contienen elementos que son necesarios para el despliegue deseado de las páginas, es por eso, que este recurso debe cargarse antes del contenido de la página.
* Las hojas de estilo usadas en todo el aplicativo tendrán que ser agregadas al template principal utilizando:

<h:head>

<h:outputStylesheet library="css" name="style.css" />

</h:head>

* Y así mismo en este template se debe crear una sección para que hojas de estilo particulares puedan ser agregadas de acuerdo a la sección que lo necesite usando:

<h:head>

<ui:insert name="head"></ui:insert>

</h:head>

* Cuando se requiera agregar una hoja de estilo particular se deberá hacer uso de esta sección utilizando:

<ui:define name="head">

    <h:outputStylesheet library="css" name="css/style.css"/>

</ui:define>

<ui:define name="content"> contenido </ui:composition>

### Colocar los Scripts al final de la página (Footer)

* El mejor lugar para colocar los scripts es al final de la página, así los contenidos no son bloqueados al generarse y son cargados lo antes posible.
* Los scripts utilizados en todo el aplicativo tendrán que ser agregados al final del template principal utilizando:

<h:body>

<ui:insert name="content"> contenido… </ui:insert>

<h:outputScript library="javascript" name="script.js" />

</h:body>

* Y así mismo en este template se debe crear una sección para que los scripts particulares puedan ser agregados de acuerdo a la sección que lo necesite usando:

<h:body>

    <ui:insert name="script"></ui:insert>

</h:body>

* Cuando se requiera agregar un script particular se deberá hacer uso de esta sección utilizando la cual deberá ser agregada al final del documento justo antes del cierre de la etiqueta:

<ui:define name="content"> contenido </ui:composition>

<ui:define name="script">

    <h:outputScript library="javascript" name="scriptParticular.js"/>

</ui:define>

### Optimizar las imágenes

Con las imágenes se debe tener en cuenta tres aspectos: tamaño, formato y el atributo de src.

Una imagen muy grande tarda mucho tiempo en cargar, es importante mantener las imágenes tan pequeñas como sea posible.

* Eliminar los metadatos de las imágenes
* Optimizar la imagen con el ancho y alto específico para no utilizar las propiedades height y width.
* Para agregar una imagen se recomienda el uso de la etiqueta:

<h:graphicImage value="/resources/images/img.png" />

* Atributo value

Verificar que la ruta en el atributo value de la imagen a cargar sea correcta.

* Formatos

**JPGE** es la mejor opción.

**PNG** Es buena opción pero puede que no haya soporte por los navegadores antiguos.

* No es recomendable usar formatos **BMPs** o **TIFFs** ya que este tipo de imágenes son muy grandes.

### Alternativa de carga de imágenes

Para conexiones lentas se puede hacer uso de imágenes directamente del documento html a través de Data URIs usando el tag: data:[<mediatype>][;base64],<data>, con esto el navegador interpreta la información y despliega la imagen sin necesidad de realizar peticiones al servidor o descargarlas lo cual incrementa la velocidad de carga.

### Despliegue de animaciones

* Utilizar mayor cantidad de código nativo posible, ya que permite que los procesos se generen más rápido.
* Ocupar transiciones y animaciones CSS3, que son más rápidas en comparación animaciones a través de Javascript.

# Integración de componentes JSF

**Descripción:** En esta sección se dará conocer las mejores prácticas en la integración de componentes para páginas web responsivas utilizando componentes JSF, Primefaces e integración con Bootstrap.



**Objetivo:** El objetivo de esta sección dar los lineamientos y buenas prácticas para integrar componentes JSF.

### Usar Controles JSF y evitar componentes de  PrimeFaces a medida de lo posible.

Muchos de los componentes de Primefaces, pueden ser reemplazados por componentes de JSF puros, obteniendo la misma funcionalidad, por lo tanto, se sugiere utilizar componentes Primefaces solo en caso de ser necesarios por la funcionalidad que estos proveen.

### Cambiar la propiedad “prependId” a false.

En el caso de sólo utilizar un form dentro de la página cambiar la propiedad prependId = “false” para acceder mucho más fácil a los controles y evitar anteponer el nombre del formulario

<h:form id="form" prependId="false">

### Evitar FORMs anidados.

* Separar adecuadamente la funcionalidad de los formularios y evitar a medida de lo posible incluir un formulario dentro de otro, ya que en la práctica, resulta un mal funcionamiento, como por ejemplo; pérdida de referencia a componentes, trunca peticiones ajax subsecuentes, pérdida de datos, etc.
* Verificar que al momento de realizar un include de otro componente no contenga un form.

### Agregar a todos los controles un identificador

Es recomendable agregar a todos los controles un identificador id para ubicarlos y acceder a ellos más fácilmente.

### Agregar la propiedad layout a block si utilizas h:panelGroup.

Al utilizar los controles h:panelGroup agregar la propiedad layout =“block” para que al momento del despliegue del control se muestre como un div.

<h:panelGroup id="identificador" layout="block"></h:panelGroup>

### Nombrar los componentes de forma coherente

* Al nombrar los componentes deberán ser consistentes a su funcionamiento y al contexto en el que interactúan.
* Establecer un estándar para el nombrado del identificador de los componentes.

### Reutilizar componentes.

Al momento de diseñar componentes o páginas deberá separar los que son de llenado de información con los que contienen botones o acciones, para poder reutilizarlos en otros lugares.

### Utilizar h:panelGroup para realizar render de secciones

Al realizar render de secciones utilizar h:panelGroup con la propiedad layout="block" y evitar controles como p:panel, p:panelGrid etc.

### Utilizar h:message

Incluye un tag <h:message for=”” > para cada componente. En caso de algún error o envío el mismo input de la página será re-renderizado con el mensaje de error.

### No usar Scriptlets

No use tags específicos como scriptlets, expresiones etc, esto hará que se dificulte una migración a componentes con Facelets.

# Diseño Responsivo

**Descripción:** Esta sección está dedicada a listar las mejores prácticas de diseño responsivo  al momento de generar páginas web. Siempre debemos de tener en cuenta la resolución de los diferentes dispositivos para los cuales se mostrará la información de las páginas.

Para cubrir el tema de diseño responsivo, se emplea el framework bootstrap, el cual proporciona soporte a las siguientes resoluciones:

* Hasta 480px
* De 481px a 767px
* De 768px a 979px
* De 980px a 1199px
* Igual o más de 1200px



**Objetivo:** Listar las mejores prácticas para un diseño web responsivo utilizando bootstrap.

### Diseñar por porcentajes, no por pixeles.

Ya no es posible seguir ocupando contenedores de tamaño único distribuidas en una pantalla de ancho fijo. El diseño responsivo obliga a la utilización de una cuadricula  flexible, donde los elementos agrupados en ella serán medidos por porcentaje de pantalla y no por pixeles.

### Diseñar para pantallas, no para dispositivos.

El mercado tecnológico actual es multidispositivo, sin estándares de ningún tipo. Por ello, la clave es pensar en todos los tamaños de pantalla, no sólo en dispositivos, lo que dejaría al diseño obsoleto en poco tiempo.

### Diseñar para tocar, no para clicks.

La interacción con los contenidos web se vuelve compleja al eliminar el mouse en las pantallas táctiles. Por eso, un diseño responsivo también debe contemplar el dedo como una entrada, lo que implica mayores tamaños en botones y links, entre otras soluciones.

### Diseñar para pantallas pequeñas primero.

El concepto clave es diseñar en resolución de pantallas desde lo más pequeño y esencial, a lo más grande y accesible.

### Los usuarios móviles merecen la misma calidad y experiencia en la navegación.



Uno de los principales principios a tener en mente cuando se están creando diseños responsivos para sitios web es asegurarse que el sitio esté construido de manera que la experiencia de navegación es la misma para todos los usuarios que la visitarán.

Esto significa que la apariencia de tu sitio y la estructura visual deberían cambiar sin generar una pérdida de contenido y funciones  para los usuarios de cualquier dispositivo o pantalla en específico. Alguien que visita desde una computadora debería tener la misma experiencia de navegación que alguien que lo visita desde un dispositivo móvil.

### Diseña tu sitio pensando en “responsivo”

Cuando estás estructurando tu sitio para convertirlo en un diseño de código real hay ciertas estructuras que son ideales para el diseño responsivo y otras que no lo son, lo que significa que hay diseños que se transformarán bien a los nuevos tamaños y otros que no gracias a su diseño y estructura.

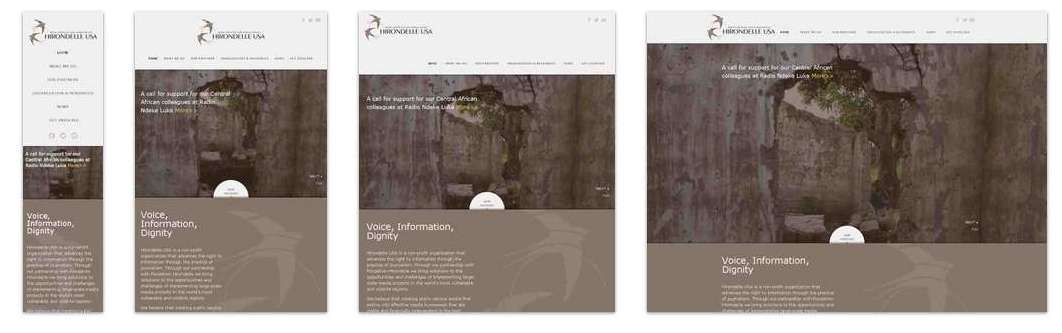
Esto quiere decir que se debe pensar en términos de simplicidad y hacer una estructura no muy compleja tanto en diseño como en código HTML y considerar los mecanismos para los elementos necesarios como la navegación y opciones de menú.

### Utilizar ems para la tipografía

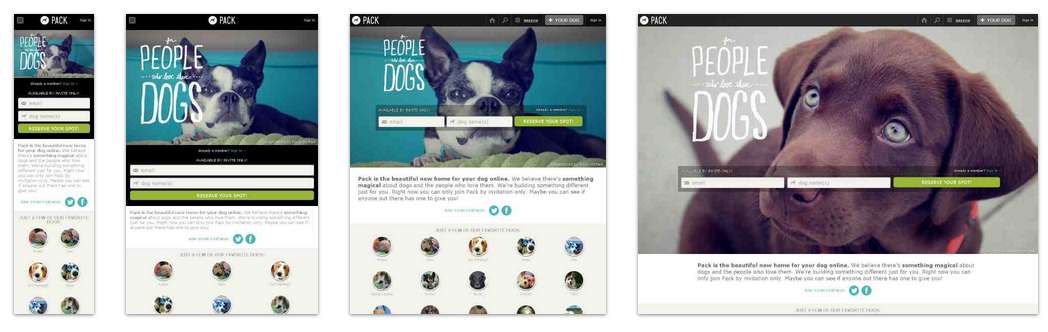
Se recomienda en la medida de lo posible utilizar “ems” en lugar de píxeles para la tipografía. Los “ems” son una unidad que toma como referencia el valor preestablecido por el navegador. Así, si el navegador viene por defecto con un tamaño de fuente de 15px, 1 em equivaldrá a estos 15px. Suponiendo a personas con problemas visuales que han cambiado el tamaño preestablecido de fuente en el navegador a 20px. Entonces 1em equivaldrá a esos 20px, siendo así una unidad flexible.

### Las [imágenes](http://www.esandra.com/out/graphicriver) en móvil y grandes resoluciones

En diseño web responsivo hay un problema llamado dirección de arte. Significa que en una vista una [imagen](http://www.esandra.com/out/graphicriver) determinada puede quedar fantástica, pero en otra se puede ver muy pequeñas sin mostrar lo que originalmente se quería representar.



Por ejemplo, la cueva de la imagen se ve muy bien en pantallas grandes, pero en el móvil pierde toda su gracia. En estos casos se habría de jugar con imágenes distintas para cada vista, tal y como vemos en el siguiente ejemplo:



### Imágenes en dispositivos Retina Display

En dispositivos móviles con pantallas de alta definición (Retina display) las imágenes usadas para la iconografía (PNG) se muestran pixeladas, para evitar este tipo de situaciones, se recomienda utilizar el formato de imágenes SVG (Gráficos Vectoriales Redimensionables), que como su nombre lo dice, son imágenes en vectores que permiten aumentar o reducir el tamaño sin comprometer la calidad de la visualización.

En comparación con los formatos JPG o PNG, el SVG tiene un peso mucho menor y puede ser escalado a cualquier tamaño sin perder la calidad de imagen, haciéndolo ideal para dispositivos con mayor densidad de pixeles por pulgada.

### Compatibilidad de SVG en navegadores antiguos

El formato SVG no es soportado por todos los navegadores,  particularmente antiguas versiones de Android(2.3 y anteriores) y Explorer (8 y anteriores), por lo cual se recomienda la combinación de  Jquery, atributos personalizados de HTML5 y [Modernizr.js](http://modernizr.com/) para así detectar si el navegador soporta imágenes en SVG. Si la aplicación no será capaz de visualizar este archivo, utilizar el atributo fallback de la etiqueta <img> para desplegar un formato de imagen compatible con el navegador. Con esta opción será necesario contar con los dos formatos para cada caso.

# Referencias

Paul Irish, Nick Cooley, Adam McIntyre, Rob Larsen, Joel Oliviera, Jared Williams, Annette Arabasz, Rob Cherny.

2014 Isobar

<http://isobar-idev.github.io/code-standards/>

Envato Pty Ltd.

2015

<http://code.tutsplus.com/tutorials/30-html-best-practices-for-beginners--net-4957>

Jean.

marzo 2010

<http://www.catswhocode.com/blog/top-10-best-practices-for-front-end-web-developers>

Scott Granneman.

1997-2015

<http://www.granneman.com/webdev/coding/formatting-and-indenting-your-html/>

W3C.

1994-2012

<http://validator.w3.org/>

[Jorge Codina](http://www.ida.cl/blog/autores/jorge/), Ideas Digitales Aplicadas SpA.

30 agosto 2013

<http://www.ida.cl/blog/desarrollo/5-soluciones-practicas-para-el-desarrollo-de-responsive-web-design/>

W3Schools

[1999-2015](http://www.w3schools.com/about/about_copyright.asp)

<http://www.w3schools.com/bootstrap/default.asp>

Cochran, David

12 de noviembre de 2012

<http://es.wikipedia.org/wiki/Twitter_Bootstrap>

FuelDeck

10 de octubre de 2013

<http://www.fueldeck.com/blog/25-resources-for-best-practices-in-web-development-2013/>

[Kathryn Aragon](http://blog.crazyegg.com/author/kathryn-aragon/)

2011-2014

<http://blog.crazyegg.com/2013/12/11/speed-up-your-website/>

High Performance Web Sites

Essential Knowledge for Front-End Engineers

Publisher: O'Reilly Media

# Firmas de Elaboración y Revisión

|  |  |  |
| --- | --- | --- |
| **Entregable** | **Clave** | **Medio** |
|  | Guía de Mejores Prácticas para el Desarrollo Web.docx | Electrónico |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Elaborado por** | **Puesto** | **Nombre** | **Firma** | **Fecha** |
|  | Ingeniero de Desarrollo | Francisco Manuel Carrasco L. |  | 26/02/2015 |
|  | Analista | Isidro López Sosa |  | 26/02/2015 |
|  | Ingeniero de Desarrollo | Jesús Martín Martínez Soto |  | 26/02/2015 |
|  | Diseñador Web | Sergio Velázquez Sánchez |  | 26/02/2015 |
|  | Arquitecto Sistemas | José Luis Méndez Santos |  | 26/02/2015 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revisión** | **Puesto** | **Nombre** | **Firma** | **Fecha** |
|  |  |  |  |  |
|  |  |  |  |  |