Universidad de las Ciencias Informáticas
Vertex, Centro de Entornos Interactivos 3D , Facultad 4

Generador de plantillas para la interfaz marcaria de la Universidad de las Ciencias Informáticas en aplicaciones web

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Frank Antonio López Acosta

Tutores: Ing. Reinaldo García Maturell

Ing. Yalena Aileth Venega Cañizares

Ing. Rodobaldo Liván Saroza Ramírez

La Habana, 2018

El primer paso es establecer que algo es posible; entonces la probabilidad ocurrirá. Elon Musk.

Dedicatoria

A mis padres, mi esposa y a mi bello y amado hijo. A mis hermanas y amigos de todas partes del mundo.

Agradecimientos

Agradecer a mi familia. Mis hermanos y amigos.

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales sobre esta, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Frank Antonio López Acosta
Autor

Ing. Reinaldo García Maturell
Tutor

Ing. Yalena Aileth Venega Cañizares
Tutor

Ing. Rodobaldo Liván Saroza Ramírez
Tutor

Resumen

La Universidad de las Ciencias Informáticas cuenta con un sistema de identidad marcaria basado en cinco marcas fundamentales, las cuales engloban productos de software afines a un objetivo en específico. En base a ello se desarrolló un sistema web que contiene las herramientas y manuales necesarios para la adaptación del software en implementación a la identidad marcaria. En la presente investigación se pretende desarrollar un generador de interfaces web que contenga componentes asociados a la identidad marcaria de los productos de la Universidad. El mismo está diseñado en base a la metodología de desarrollo de programación extrema e implementado en una arquitectura basada en componentes. Las tecnologías a utilizar estarán centradas en el *framework* de presentación Bootstrap 4.0, con componentes web encapsulados con programación TypeScripts y JavaScript. Finalmente, se realiza la validación del sistema, mediante pruebas de aceptación como método principal que propone la metodología de desarrollo.

Palabras clave: generador, identidad marcaria, plantillas, web.

Índice general

Índice de figuras

Índice de tablas

Introducción

El acelerado desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) , juega un papel protagonista en muchas de las esferas de la vida cotidiana del hombre. Estas han transformado la manera de trabajar y gestionar los recursos, convirtiéndose en un elemento clave para que la labor del hombre sea más productiva.

Dentro de las TIC, la industria del software alcanza una posición privilegiada por su capacidad de controlar o hacer accesibles los adelantos electrónicos. La industria de las TIC está sujeta al desarrollo económico de cada nación. Cuba no esta exenta de estos adelantos tecnológicos, ya que realiza importantes acciones en torno a las nuevas tecnologías, debido a que esta, es una industria que marca y determina las pautas para el desarrollo futuro del país.

La Universidad de las Ciencias Informáticas (UCI) como parte de su objetivo fundamental para lograr la informatización de la sociedad cubana, ha desarrollado diferentes productos de software enfocados cada uno en las diferentes ramas de la economía del país. Todos estos productos necesitan ser ajustados a una identidad donde sea posible categorizarlos, catalogarlos e identificarlos por el cliente o la institución que los vaya a utilizar.

Para lograr una estandarización en la identificación de los productos, logotipos o estilos, la universidad propone una solución basada en cinco marcas fundamentales: Xavia, Xauce, Xilema, Xedro y Xabal, las cuales difieren en cuanto a temática y diseño. Xavia engloba todos los productos asociados a la rama de la salud. Xauce incluye los productos referidos a sistemas de educación. Xilema aborda los software referidos a los servicios telemáticos. Xedro contiene los productos dirigidos a empresas e industrias. Mientras que Xabal posee como línea de productos, todo lo referido a temáticas de administración pública.

A partir de lo anteriormente planteado, se desarrolló el sistema web Estrategia Marcaria de los productos UCI que facilita la modificación estética de elementos asociados al producto, tales como, un generador de logotipos y soportes promocionales. Todos los productos desarrollados en la universidad, ya sean aplicaciones web o de escritorio, deben utilizar dicha identidad marcaria. De esta forma, se logra una línea de estilo unificada, regida bajo las normas establecidas para los productos desarrollados por la UCI. Este sistema cuenta las siguientes limitaciones:

- Dentro del sistema, se encuentran disponibles ejemplos de las interfaces que cumplen con los estándares establecidos para las marcas. La tecnología utilizada para la implementación de estos prototipos no se ajusta a las tendencias actuales y presenta problemas de compatibilidad con la línea de desarrollo

que se está llevando a cabo en la universidad. Esto trae consigo, que en el caso de las aplicaciones web existan problemas de compatibilidad, ya que los elementos que conforman estas interfaces se desarrollaron en la versión 2.0 del *framework* Bootstrap. Los nuevos proyectos a desarrollar por los centros de producción, realizan una actualización de la tecnología a utilizar por lo que no resulta concordante con el sistema actual de identidad marcaria.

- Provee un conjunto de manuales distribuidos por cada una de las marcas, donde se explican las pautas de diseño de cada uno de los componentes que deben formar parte del producto final. Esta práctica dificulta el trabajo de diseño web, puesto que los manuales son extensos y de complejo entendimiento para usuarios con poca práctica en esta rama.
- Los prototipos de ejemplos expuestos en el sistema, no permiten obtener el código fuente de las plantillas predefinidas. Esto dificulta al equipo de desarrollo obtener dichas plantillas para su uso posterior, teniendo que diseñar manualmente las interfaces para sus proyectos, lo que conlleva un mayor esfuerzo por parte de los desarrolladores.

Teniendo en cuenta la situación problemática anteriormente descrita, se define como **diseño teórico de la investigación**:

- **Problema de la investigación**: ¿Cómo contribuir a la generación de interfaces, para el diseño de aplicaciones web, que apliquen la identidad marcaria de la Universidad de las Ciencias Informáticas?
- **Objeto de estudio**: Generación de interfaces gráficas en el desarrollo de aplicaciones web.
- **Objetivo general**: Desarrollar un sistema que permita la creación de interfaces web que hagan uso de la identidad marcaria de la Universidad de las Ciencias Informáticas.
- **Campo de acción**: Generación de interfaces gráficas en el desarrollo de aplicaciones web basadas en identidades marcarias de la Universidad de las Ciencias Informáticas.

Para cumplir con las exigencias de la investigación, se definen un conjunto de tareas que marcan el camino de la misma. Estas se exponen a continuación:

Tareas de la investigación:

1. Elaboración del marco teórico de los principales conceptos asociados a la generación de interfaces para aplicaciones web.
2. Identificación de las principales tecnologías para la generación de interfaces para aplicaciones web.
3. Realización del análisis y diseño de la aplicación.
4. Implementación de un Generador de plantillas web que permita la creación de interfaces haciendo uso de la identidad marcaria UCI.
5. Validación de la herramienta propuesta a través de los métodos definidos en la investigación.

En la investigación se destaca la utilización de los siguientes métodos teóricos:

- **Analítico-sintético**: Se utiliza este método científico de investigación para poder descomponer el problema en partes más concretas, llegar a conclusiones más específicas y lograr la correcta integración y comunicación entre las mismas. La investigación, partirá del análisis de la Web 2.0 y los sistemas gestores de contenidos, como base para lograr el entendimiento de los generadores de interfaces web.
- **Análisis histórico-lógico**: El mismo fue seleccionado para llevar a cabo un análisis valorativo de la bibliografía existente, así como para conocer las tendencias actuales en cuanto a la generación de interfaces web. Se analiza el estado del arte existente respecto a la problemática planteada, se selecciona la bibliografía así como las herramientas y tecnologías más utilizadas para este fin.
- **Inductivo-deductivo**: El cual permite reflejar los elementos comunes entre las temáticas estudiadas, establecer generalizaciones y analizar cada uno de los detalles hasta establecer las relaciones existentes entre los mismos. Para ello, se realiza el estudio de los sistemas homólogos que basan su negocio en la generación de interfaces web.
- **Modelación**: Utilizado para elaborar los diferentes modelos definidos en la metodología escogida y que sirven de guía durante el desarrollo del generador de interfaces web.

Los métodos empíricos utilizados para obtener información sobre el objeto de estudio fueron:

- **Consulta bibliográfica**: Permite la elaboración del marco teórico de la investigación fundamentada por la información consultada referente a conceptos asociados a la generación de interfaces web.
- **Observación científica**: Utilizado con el objetivo de conocer cómo están estructurados los sistemas generadores de interfaces web utilizados actualmente.
- **Pruebas de validación**: Confirma la veracidad y utilidad del generador de interfaces web. Para ello, se realizan las pruebas que propone la metodología de desarrollo seleccionada, como las pruebas de aceptación.

El trabajo de diploma se estructura de la siguiente forma:

Capítulo 1: **Fundamentación teórica**:
En este capítulo se hace referencia a los conceptos y definiciones de la investigación. Se realiza el estudio de los sistemas homólogos generadores de interfaces web, además de llevar a cabo la definición de las herramientas y tecnologías a usar, así como la selección de la metodología de desarrollo de software.

Capítulo 2: **Características y diseño del sistema**:
En dicho capítulo además de una descripción de la solución propuesta, se detalla el análisis y diseño de la misma, definiendo los requerimientos identificados con el cliente. Además se especifican los distintos patrones de diseño y el patrón arquitectónico a utilizar.

Capítulo 3: **Implementación y prueba del sistema**:
El capítulo hace alusión en una primera parte al proceso de implementación de la herramienta, reflejando

el cumplimiento de los requerimientos funcionales propuestos, para luego describir el proceso de prueba de cada una de las funcionalidades implementadas a partir del diseño de casos de pruebas.

Por último, se exponen las conclusiones y recomendaciones derivadas de la investigación, las referencias bibliográficas, así como los anexos que apoyan la comprensión y dan información adicional sobre el trabajo realizado.

CAPÍTULO 1

Fundamentación teórica

1.1. Introducción

En el presente capítulo, se lleva a cabo un estudio sobre los conceptos más importantes referentes al tema de la investigación. Se abordaron los principales conceptos y definiciones de la Web 2.0 y los Sistemas Gestores de Contenidos (CMS), para lograr definir la concepción de los generadores de interfaces web. Se realiza un estudio de los sistemas homólogos que basan su negocio en el problema de la investigación y también, se definen las herramientas y tecnologías que serán utilizadas en la solución. Además, se selecciona la metodología de software para el desarrollo.

1.2. ¿Qué es la Web 2.0?

La Web 2.0 es la transición que se ha dado de aplicaciones tradicionales, hacia aplicaciones que funcionan a través de la web, enfocadas al usuario final. Se trata de aplicaciones que generen colaboración y de servicios que reemplacen las aplicaciones de escritorio. Básicamente se ha reinventado lo que era el Internet, y es que cuando la web inició, se encontraba en un entorno estático, con páginas que sufrían actualizaciones y no tenían interacción con el usuario (Van Der Henst, 2005).

El término Web 2.0 está asociado a aplicaciones web que facilitan el compartir información, la interoperabilidad, el diseño centrado en el usuario y la colaboración en la World Wide Web (WWW, por sus siglas en inglés). Un sitio web 2.0 permite a los usuarios interactuar y colaborar entre sí, como creadores de contenido, generado por usuarios en una comunidad virtual, a diferencia de otros donde los usuarios se limitan a la observación pasiva de los contenidos que se han creado para ellos (ROMANÍ y KUKLINSKI, 2008).

Esta evolución en la web pasa al usuario a un primer plano en cuanto al enfoque del software, logrando a través de Internet proveer servicios alojados en la web. Las redes sociales y blogs son medios por los cuales,

los internautas aportan contenido a Internet y facilitan la creación de los mismos, aportando interoperabilidad y logrando compartir información. Por otra parte, otorga la posibilidad de acceder a herramientas útiles sin necesidad de tenerlas instaladas en su ordenador.

La Web 2.0 se caracteriza por la participación del usuario como contribuidor activo y no solo como espectador de los contenidos de la Web (usuario pasivo). Según (Vallés, 2011), los tipos de aplicaciones que componen este nuevo estándar en Internet son las siguientes:

- Blogs colaborativos.
- Redes sociales.
- Sistemas Gestores de Contenidos (CMS).
- Aplicaciones web dinámicas.

Entre las aplicaciones mostradas anteriormente, se analizarán los CMS como aplicaciones que permiten la creación y administración de contenidos por medio de páginas web. Esto lograría una mejor comprensión de la fundamentación teórica, logrando situar estos conceptos en el tema de la investigación.

1.2.1. Sistemas de gestión de contenidos (CMS)

Desde el año 2000 se ha producido una convergencia entre las plataformas, de forma que pueden encontrarse actualmente soluciones que pretenden ser globales y ofrecer soporte a todo el proceso de gestión de información en una organización. Las herramientas para este trabajo han recibido la denominación de CMS, y se han integrado con los sistemas de gestión documental y con los de recuperación de información (Tramullas y Garrido, 2006).

Un CMS es una herramienta que permite a un editor crear, clasificar y publicar cualquier tipo de información en una página web. Los CMS trabajan de modo tal que el editor actualiza una base de datos, incluyendo nueva información o editando la existente, según (Cuerda y Minguillón, 2004). Por lo tanto, el gestor de contenido es una aplicación informática usada para crear, editar, gestionar y publicar contenido digital multimedia en diversos formatos. El gestor de contenidos genera páginas web dinámicas interactuando con el servidor web para generar la página web bajo petición del usuario, con el formato predefinido y el contenido extraído de la base de datos del servidor.

Esto permite gestionar, bajo un formato estandarizado, la información del servidor reduciendo el costo de gestión del portal con respecto a un sitio web estático, en el que cada cambio de diseño debe ser realizado en todas las páginas web, de la misma forma que cada vez que se agrega contenido tiene que maquetarse una nueva página HyperText Markup Language (HTML, por sus siglas en inglés) y subirla al servidor web. Bajo este concepto, es posible definir los generadores de interfaces web, ya que se han abordado aspectos esenciales para comprender su procedencia y concepción.

1.2.2. ¿Qué es un generador de interfaces web?

Un generador de interfaces web es una versión simplificada de los CMS. Mientras que los sistemas clásicos de gestión de contenido se usan para el mantenimiento y gestión de grandes páginas web, los generadores de interfaces web están pensados para proyectos web, en los cuales el diseñador del proyecto no tiene que tener conocimientos de programación, logrando alcanzar una interfaz web funcional (J. García, 2014).

El generador de interfaces o software generador de interfaces es una herramienta para desarrollar páginas web, correos electrónicos y plantillas para documentos, sin tener que hacer el formato manualmente o escribir lenguaje mediante códigos de programación. Esta herramienta proporciona una interfaz gráfica del usuario, Graphical user interface (GUI, por sus siglas en inglés) para fines de diseño y produce una fuente de código o formato estructurado para páginas web, correos electrónicos o documentos (Quevedo Ortiz et al., 2013).

El generador de interfaces es una herramienta, para desarrollar páginas web y plantillas para documentos sin tener que hacer el formato manualmente o escribir lenguaje mediante códigos de programación. Esta herramienta proporciona una interfaz gráfica del usuario para fines de diseño y produce una fuente de código o formato estructurado para páginas web (Bolagay y Andrés, 2015).

Por lo expuesto anteriormente, se llega a la conclusión, de que un generador de interfaces web es una herramienta capaz de brindar la posibilidad de crear y diseñar un prototipo web sin poseer conocimientos previos de programación, a través de una interfaz amigable y de fácil interacción para el usuario. En el caso del objeto de estudio, es imprescindible contar con componentes asociados a la identidad marcaria de los productos UCI.

1.2.3. Tipos de generadores web

Existen generadores de páginas web que están alojados en el sistema del proveedor que los ofrece y, por otra parte, existen también generadores de páginas web que deben instalarse como software en un ordenador. Ambas variantes presentan ventajas e inconvenientes: mientras que la opción de instalación suele tener un funcionamiento más rápido; la opción de hosting online (alojamiento en línea) implica que puede acceder en cualquier momento desde cualquier lugar. Con el rápido incremento y mejora del acceso a Internet, las herramientas en línea están ganando popularidad, esta es la razón por la que el estudio de los sistemas homólogos se centra en las soluciones en línea (J. García, 2014).

1.3. Sistemas homólogos estudiados

Para realizar el estudio de los sistemas homólogos se visitaron varios sitios teniendo en cuenta diferentes características. Estos sitios especializados son los más utilizados a nivel internacional según *Google Trends*, herramienta con la cual es posible determinar las actuales tendencias de este negocio en Internet, así como el nivel de usuarios que acuden a estas. Para profundizar en el impacto de este tipo de herramienta en particular, se analizarán las estadísticas del mismo en cuanto a búsquedas globales por países referentes al término de **generadores de interfaces web**, tornándose de color azul los países con más búsquedas en estos términos.

Figura 1.1. Búsqueda de generadores web por países (Fuente: Google Trends).

En la imagen que se muestra a continuación, se puede apreciar el número de búsquedas del término **generadores de interfaces web** en cuanto al tiempo.

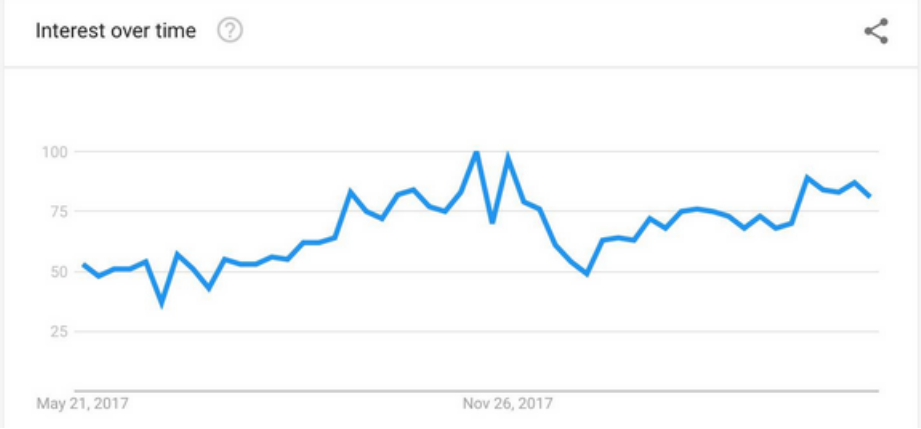


Figura 1.2. Búsqueda de generadores web en línea de tiempo (Fuente: Google Trends).

Al analizar las figuras anteriores se puede constatar que este tipo de herramientas es popular en países con alto desarrollo tecnológico y está siendo una de las opciones más buscadas debido a las ventajas que ofrecen. A continuación, se tomaron muestras de estas búsquedas y se seleccionaron las seis herramientas

generadores de interfaces web más populares.

1. Adobe Dreamweaver (www.dreamweaver.com)

Adobe Dreamweaver es un editor que permite crear páginas compatibles con teléfonos móviles y tabletas, además de sitios web tradicionales, gracias a sus avanzadas funciones *responsive*. Según (PASCUAL, 2008), cuenta con las herramientas fundamentales para todas las tecnologías utilizadas actualmente en la web y se puede actualizar en línea.

La ventaja de este editor, radica en las posibilidades que brinda de ampliación y personalización, puesto que en este programa sus rutinas (como la de insertar un hipervínculo, una imagen o añadir un comportamiento) están hechas en Javascript-C, lo que le ofrece flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++, sino rutinas de Javascript que hace que sea un programa fluido. Todo ello permite que programadores y editores web hagan extensiones para su programa y conformen su diseño.

2. Incomedia WebSite (www.incomediawebsite.com)

WebSite propone una interfaz renovada y añade funciones enfocadas en el terreno del diseño web. Según (Montañés, 2011), las páginas generadas con WebSite son *responsive* y se adaptan a cualquier dispositivo, ya sean computadoras de escritorio, portátiles, móviles o tabletas. La sencillez de uso, es una de sus características destacables, con la posibilidad de crear desde páginas personales o blogs hasta tiendas para comercio electrónico seguros. El alojamiento es otro de los servicios ofrecidos por esta herramienta.

Ofrece adaptabilidad a los diferentes perfiles de usuario, incluyendo a aquellos que no tienen ningún conocimiento de programación o diseño web y el perfeccionamiento de la interfaz de usuario enriquecida. También, destacan la plena compatibilidad con HTML5 que garantiza la adaptación a cualquier tipo de dispositivo móvil y la cuidada introducción de las redes sociales

3. Avanquest WebEASY Professional (www.avanquestwebeasy.com)

WebEasy se adapta a los estándares actuales de la web y proporciona más de 1.000 combinaciones con sus plantillas incluidas. Según (O. García, 2009), este software cuenta con una interfaz WYSIWYG y permite la edición de HTML. Ofrece más de 600 plantillas para personalizar y es una aplicación de diseño web funcional y fácil de usar, con módulos integrados para Google Maps®, YouTube®, Facebook®, Flickr® y numerosas aplicaciones sociales.

En las partes negativas, es un programa simple y con una interfaz exclusiva. Sobre todo, a la comunidad de desarrolladores puede resultarle complejo para sus implementaciones, porque no ofrece ninguna utilidad que permita a los usuarios que sí tienen conocimientos de HTML y otros lenguajes mantener un control adicional sobre el resultado final (O. García, 2009).

4. Magix Web Designer (www.magixwebds.com)

Con dos productos en su gama de programas de diseño web, el fabricante alemán Magix cuenta con una gran reputación para la creación de productos multimedia. Magix Web Desingner y su versión Premium incluyen galerías web y elementos de diseño para facilitar el proceso de creación. Según (Onieva García, 2010) la meta de este software es conseguir generar una web lista en poco tiempo. Sus funciones incluyen edición básica de imágenes, adaptación a distintos formatos de dispositivos o integración con archivos multimedia y redes sociales.

Se compone la página de forma intuitiva y cada objeto se puede mover, trasladar, girar y editar propiedades. Se puede trabajar con formas, capas (que se pueden ordenar por profundidad) y maneja transparencias. Además, puedes exportar el resultado a HTML, con las correspondientes hojas de estilo y scripts.

5. Google Sites (www.googlesites.com)

Google Sites, es el creador de sitios de Google, permite la creación, modificación y extensión de páginas web. Según (Bailén y Bernabeu, 2011), Google Sites se ha convertido en una plataforma popular para crear sitios de intranet y sitios escolares. Es una solución gratuita y ofrece algo más que crear páginas web. Con opciones para organizar reuniones, compartir información y colaborar en proyectos.

Cuenta con ventajas significativas tales como que no requiere programación como el HTML o Cascading Stylesheets (CSS, por sus siglas en inglés). Aunque se puede editar directamente parte del código, la integración de contenidos no requiere contar con estos conocimientos. Dispone de plantillas y fácil creación de las mismas. Provee fácil manejo de archivos adjuntos a través de Google Drive y fácil integración de contenido multimedia (vídeos, documentos, hojas de cálculo y presentaciones del ambiente Google Docs, Google Fotos y herramientas de iGoogle).

6. Axure RP Pro (www.Axure.com)

Es una herramienta que está dirigida tanto a la creación de aplicaciones web como de escritorio. Es fácil comenzar a trabajar con Axure, ya que es intuitiva y es posible comenzar a producir sin necesidad de seguir un manual. Una de sus características fundamentales, es que genera prototipos HTML de sitios web sin necesidad de ningún tipo de codificación (Axure, 2012).

El software está disponible en el mercado en dos versiones: *Standard* y Axure RP Pro. Ambas versiones

son de pago, pero es posible obtener una versión prueba por 30 días.

7. Wix (www.wix.com)

Wix.com es una plataforma para el desarrollo web basada en la nube que fue desarrollada y popularizada por la compañía Wix. Según (Padilla Carrascal, 2016) permite a los usuarios crear sitios web HTML5 y sitios móviles a través del uso de herramientas de arrastrar y soltar en línea. Los usuarios pueden agregar funcionalidades como plug-ins, e-commerce, formularios de contacto, marketing por correo electrónico, y foros comunitarios con sus sitios web utilizando aplicaciones desarrolladas por Wix y de terceros.

Wix es construido en un modelo de negocio *freemium*, ganando sus ingresos a través de actualizaciones *premium*. Los usuarios deben comprar paquetes *premium* para conectar sus sitios a sus propios dominios, eliminar los anuncios Wix, añadir capacidades de comercio electrónico o comprar almacenamientos de datos y ancho de banda adicionales.

1.3.1. Conclusiones del estudio

En el estudio realizado sobre generadores de sistemas web se analizaron diversas características importantes que deberían estar presentes dentro de estos sistemas homólogos. A continuación, se muestra una tabla comparativa con los resultados, en la cual se comparan estos sistemas dependiendo de características previamente definidas tales como las posibilidades de obtener una interfaz adaptativa, si es un software libre o de pago, la capacidad de integración con otras herramientas que formen parte del hilo de desarrollo, así como si es un sistema web o de escritorio.

Aplicaciones	Pantalla adaptativa	Software Libre	Integración	Sistema web	Gratuita
Adobe Dreamweaver CC 2017	SI	NO	SI	NO	NO
Incomedia WebSite X5 v14	SI	NO	NO	NO	NO
Avanquest WebEASY Professional	SI	NO	NO	NO	NO
MagixWeb Designer	SI	NO	SI	NO	NO
Google Sites	SI	NO	SI	SI	NO
Axure RP Pro	SI	NO	SI	NO	NO
Wix	SI	NO	SI	SI	SI

Figura 1.3. Comparativa entre sistemas homólogos (Fuente: Elaboración propia).

El estudio realizado sobre los sistemas generadores de interfaces web, destaca que los mismos son privativos (no existen variantes de código abierto que cuenten con soporte y disponibilidad en el panorama actual) por lo que se hace necesario realizar un pago previo para acceder a la totalidad de sus funcionalidades. No se tiene acceso al código fuente, ya que las interfaces generadas son hospedadas dentro de la misma herramienta, por lo que también los datos e información se alojan fuera del control del usuario. En otros casos, se ajustan solo a las necesidades y características de las instituciones que los desarrollaron, o no están orientados a todo tipo de usuarios.

El sistema que cumple con la mayoría de las condiciones necesarias para la investigación es Wix. Es el más buscado por países como muestra la siguiente figura:

Figura 1.4. Búsqueda del término Wix por países (Fuente: Google Trends).

Además, es el más buscado en comparación con los otros sistemas homólogos, como muestra la siguiente figura. El término Wix es representado por la línea de tiempo de color **azul**, Axure RP Pro representado por la línea **roja** y los restantes por la línea de color **verde** y **violeta**.

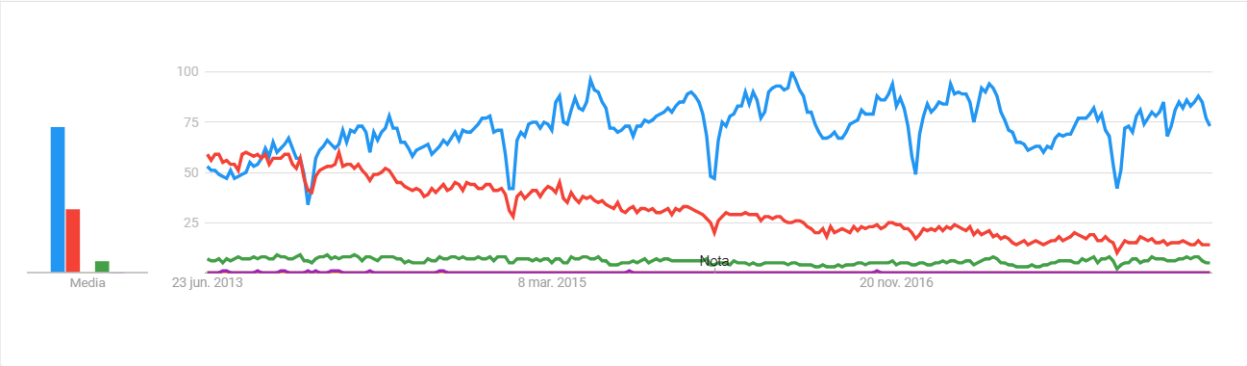


Figura 1.5. Comparación de búsquedas del término Wix en línea de tiempo (Fuente: Google Trends).

Este sistema está entre los mejores disponibles en la web siendo el generador web más utilizado y buscado por la mayoría de los países y en pleno ascenso en cuanto a consultas mensuales. La opción de cuenta gratis le permite a los usuarios crear y hospedar profesionales sitios web sin costo. No se necesita saber codificar o programar para diseñar un sitio web personalizado, debido a que sus herramientas interactivas facilitan el trabajo. Y el nivel totalmente gratuito te permite construir y hospedar un sitio indefinidamente sin costo.

Sin embargo, ofrece desventajas significativas tales como que no existe manera de ver las estadísticas del sitio generado. El paquete gratuito de Wix, trae consigo la presencia de anuncios publicitarios, si bien, es común en muchos de los generadores web, en Wix los mismos, cuentan con un tamaño prominente y una política agresiva de publicidad.

En conclusión, este sistema no solucionaría el problema de la investigación debido a que provee acceso a un gran número de funcionalidades gratuitas, sin embargo, para acceder a la totalidad de sus funcionalidades se necesitaría ser usuario *Premium* mediante un pago. Además, no es un sistema de software libre, lo cual es un aspecto fundamental en la línea de desarrollo llevada a cabo en la universidad y en el país.

En cambio, el generador de interfaces web para la identidad marcaria de los productos UCI se centra en ofrecerle al desarrollador una herramienta donde pueda diseñar y realizar un prototipo de interfaz que posea todas las características estéticas asociadas a las marcas establecidas por la universidad.

A partir del estudio de los generadores web homólogos, se concluyó que estos poseen características comunes en cuanto a funcionalidades y servicios. En base a esto, se determinaron características afines con el posible diseño del generador de plantillas web para la identidad marcaria de los productos UCI.

Barras de navegación: es prácticamente constante el uso de barras de navegación ya que promueve un diseño simple y práctico, que facilita la unificación del contenido disponible para el usuario al momento de diseñar su plantilla web.

Enlace específico para generar web: es un elemento fundamental que guía al usuario directamente al generador evitando confusiones entre otros elementos contenidos en la página principal del sitio.

Exportadores de código: es una funcionalidad recurrente ya que ofrece la posibilidad al usuario de obtener el código fuente para su posterior desarrollo en caso de que utilice otras herramientas de trabajo.

Componentes web editables en cuanto al tema del diseño: ofrece la posibilidad al usuario de personalizar algunas características de los componentes web a seleccionar.

1.4. Metodología de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de técnicas, procedimientos y herramientas que ayudan a los desarrolladores a realizar un nuevo producto. Indican paso a paso todas las actividades a realizar para lograr el producto informático deseado, señalan además el papel que tendrán las personas que estarán presentes en el desarrollo de las actividades (Letelier, 2006).

El desarrollo de software es un proceso difícil y lleno de riesgos donde la metodología define quién debe hacer qué, cuándo y cómo debe hacerlo. El fracaso en el desarrollo de un sistema está estrechamente relacionado con una mala selección de la misma, por lo cual este proceso debe llevar un análisis previo y exhaustivo por parte del equipo de desarrollo (D. Gutierrez, 2011). Estas se clasifican en dos grupos:

- **Tradicionales**: Se centran en la definición detallada de los procesos y tareas a realizar, herramientas a utilizar, y requiere una extensa documentación, ya que pretende prever todo de antemano. Este tipo de metodologías son más eficaces y necesarias cuanto mayor es el proyecto que se pretende realizar respecto a tiempo y recursos que son necesarios emplear, donde una gran organización es requerida (Expósito, 2008).

- **Ágiles**: Orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del producto al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios según se va desarrollando (Pérez, 2011).

1.4.1. Metodologías tradicionales

Entre las metodologías tradicionales más populares se encuentra Rational Unified Process (RUP, por sus siglas en inglés), que centran su atención en mantener una documentación exhaustiva del proyecto y cumplir con el plan previsto y definido con precisión en la fase inicial del desarrollo del proyecto. Las metodologías tradicionales suelen enfatizar en la documentación, la planificación y el seguimiento riguroso de múltiples actividades (Tinoco Gómez; Rosales López y Salas Bacalla, 2010).

RUP

Es una metodología de desarrollo de software que intenta integrar todos los aspectos a tener en cuenta durante el ciclo de vida del sistema, con el propósito de abarcar grandes o pequeños proyectos de software. Proporciona un acercamiento a la asignación de tareas y responsabilidades en una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad, que se ajuste a las necesidades de sus usuarios finales. Provee herramientas para los pasos del desarrollo y la documentación de ayuda para sus clientes (A. Martínez y R. Martínez, 2014).

MSF

Microsoft Solutions Framework (MSF, por sus siglas en inglés) es una metodología eficaz para las organizaciones que desean desarrollar de manera rápida soluciones tecnológicas de alta calidad y relevantes para el negocio. Su flexibilidad permite adaptarlo de manera sencilla a la mayoría de proyectos tecnológicos, lo que ayuda a los equipos a comunicarse y coordinar las actividades más importantes (Turner, 2006).

1.4.2. Metodologías ágiles

Existen diversas metodologías que coinciden en llamarse metodologías ágiles, y aunque entre ellas comparten muchas características, tienen también diferencias significativas. A continuación, se presentan algunas de las metodologías ágiles más representativas.

SCRUM:

Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprint*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Éstas son las verdaderas

protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Schwaber, 2013).

eXtreme Programming (XP):

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de Extreme Programming (XP, por sus siglas en inglés), describe la filosofía de XP sin cubrir los detalles técnicos y de implantación de las prácticas. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos. A continuación se presentan las características esenciales de XP organizadas en los tres apartados siguientes: historias de usuario, roles y proceso (Letelier, 2006).

1.4.3. Selección de la metodología a utilizar

Luego de analizar las características de las metodologías mencionadas anteriormente, se selecciona XP para que guíe el proceso de desarrollo, debido a que:

- Es la que mejor se ajusta a las necesidades del proyecto en cuanto a recursos técnicos, humanos y tiempo de desarrollo.
- El equipo de trabajo es pequeño y el cliente forma parte de mismo, lo cual mitiga los riesgos y aumenta la probabilidad de éxito.
- Es adecuada para proyectos con requisitos imprecisos y cambiantes como es el caso de la investigación.
- Permite que en cualquier momento del proceso de desarrollo de la aplicación, se pueda regresar a operaciones anteriores sin afectar el ciclo de vida del software.

1.5. Herramientas y tecnologías

A continuación, se describirán las tecnologías a emplear, las cuales serán seleccionadas a partir de las ventajas que estas puedan ofrecer para conformar la solución. Se seleccionarán los lenguajes de programación a utilizar y el *framework* web. Además, en este apartado se seleccionarán las herramientas mediante las cuales se llevará a cabo la implementación de la solución. Se tendrá en cuenta el entorno de desarrollo integrado a utilizar, así como las herramientas de ingeniería de software.

1.5.1. Lenguaje de modelado

El Lenguaje Unificado de Modelado (Unified Modeling Language (UML, por sus siglas en inglés)), es un lenguaje estándar para escribir planos de software, UML se puede utilizar para visualizar, especificar, construir y documentar los artefactos para el desarrollo de un software. UML prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan (Hernández Orallo, 2000).

1.5.2. Herramienta de modelado

Visual Paradigm 8.0

Visual Paradigm es una herramienta de modelado profesional que hace uso del lenguaje UML. Una característica esencial de Visual Paradigm es que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción y despliegue. Visual Paradigm, permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Paradigm, 2013).

1.5.3. *Framework* web

Un *framework* web es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Este se puede considerar como una aplicación genérica incompleta y configurable a la que es posible añadir las últimas piezas para construir una aplicación concreta (Rittberger y Blees, 2009). Específicamente se utilizará el *framework* Bootstrap 4.0 para manejar los estilos CSS en la solución. El mismo se describe a continuación.

Twitter Bootstrap 4.0

Bootstrap es un *framework* desarrollado y liberado por Twitter que tiene como objetivo facilitar el diseño web. Permite crear interfaces web de diseño adaptable, es decir, que se ajusten a cualquier dispositivo y tamaño de pantalla. Es de código abierto, por lo que se puede usar de forma gratuita y sin restricciones (Ledesma, 2008).

1.5.4. Lenguajes de programación

Un lenguaje de programación permite crear un grupo de instrucciones que luego se convertirán en un programa (Sala, 2003). Teniendo en cuenta que el sistema de identidad marcaria de los productos UCI está desarrollado con tecnologías específicas, para el desarrollo del generador de interfaces web será utilizado HTML5, TypeScript y JavaScript como lenguajes de programación. A continuación, una breve descripción de estos lenguajes.

HTML5

Es un lenguaje markup usado para estructurar y presentar el contenido para aplicaciones web. Constituye la quinta revisión del estándar que fue creado en 1990. HTML5 está relacionado con la entrada en decadencia de anteriores estándares como HTML 4, el cual se combinaba con otros lenguajes para producir los sitios que están disponibles en la web. Este lenguaje ofrece otras posibilidades para explotar usando menos recursos, además entra en desuso el formato XHTML, dado que ya no sería necesaria su implementación (Gauchat, 2012).

JavaScript

Es uno de los más potentes e importantes lenguajes de programación en la actualidad. Entre sus principales ventajas se encuentra que es útil, práctico y está disponible en todos los navegadores web actuales. JavaScript es creado por Brendan Eich y vio la luz en el año 1995 con el nombre de LiveScript, que luego fue nombrado JavaScript, nace como un lenguaje sencillo destinado a añadir algunas características interactivas a las páginas web. Sin embargo, hoy en día ha crecido de manera acelerada y es el lenguaje de programación que se utiliza en los sitios web en el mundo.

La potencialidad del mismo la ofrece principalmente en lado *frontend*, agregando mayor interactividad a la web, también es posible usar las librerías y frameworks como: jquery, angular, backbone y react, escritas sobre JavaScript, y que ayudan a crear una mejor experiencia de usuario en los sitios web (E. Gutierrez, 2009).

Typescript

Es un lenguaje libre y de código abierto desarrollado por Microsoft que actúa como un superconjunto de Javascript. Básicamente añade una tipología estática mediante objetos basados en clases. El sistema de tipos de Typescript realiza una formalización de los tipos de Javascript, mediante una representación estática de sus tipos dinámicos. Esto permite a los desarrolladores definir variables y funciones tipadas sin perder la esencia de Javascript. Poder definir los tipos durante el tiempo de diseño nos ayuda a evitar errores en tiempo de ejecución, como podría ser pasar el tipo de variable incorrecto a una función. Esta tipología no se refleja en el código final, de hecho una interfaz, por ejemplo, no añade sobrecarga en el código final (Ramos Muñoz et al., 2017).

1.5.5. Entorno de desarrollo integrado

NetBeans 7.3.1

NetBeans es un proyecto de código abierto con una comunidad en constante crecimiento. Hoy en día hay disponibles dos productos: el Entorno de Desarrollo Integrado, Integrated Development Environment (IDE, por sus siglas en inglés) NetBeans y la Plataforma NetBeans. NetBeans es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además, un número importante de módulos para extender el IDE NetBeans. Es un producto libre y gratuito sin restricciones de uso (Oracle, 2015).

1.6. Conclusiones parciales

A partir de la fundamentación teórica y la investigación referente a las herramientas, tecnologías y metodologías, se arribaron a las siguientes conclusiones:

- Los sistemas homólogos estudiados, no cuentan con las condiciones necesarias para desarrollar la solución. Por ello, se hace necesario diseñar e implementar un generador de interfaces web que se ajuste a las necesidades de la investigación.
- La definición de estas tecnologías y herramientas, logrará conformar una propuesta de solución compatible con las líneas de desarrollo llevadas en la universidad. Esto fomenta el uso de entornos de código abierto con tecnologías actualizadas garantizando la compatibilidad de las aplicaciones en desarrollo con la solución.
- Se adopta XP como metodología de desarrollo para llevar a cabo la ingeniería de software, pues permite generar los artefactos necesarios para una mayor comprensión de la solución a desarrollar. Además, se ajusta a las necesidades de la investigación, debido a que se cuenta con un plazo de tres meses para desarrollar la solución, contando con un equipo reducido a un integrante.

CAPÍTULO 2

Características y diseño del sistema

2.1. Introducción

En este capítulo se describen las características y diseño del sistema, partiendo de la descripción de la solución. Se definen las historias de usuario. Se abordarán las fases de planificación y diseño del sistema, donde se describirán los artefactos generados por la metodología. Además, se define la arquitectura y los patrones utilizados para la solución.

2.2. Descripción de la propuesta de solución

A partir de analizar los elementos comunes entre los sistemas homólogos, se define para la propuesta de solución, que el sistema debe estar compuesto por una página principal con enlaces directos para cada una de las marcas, dándole acceso al usuario a un generador web donde estén integrados los temas de estilos asociados a cada marca en específico. El generador de interfaces web debe estar formado por un conjunto de categorías que engloben componentes web comunes y a su vez, otros que cumplan con las pautas de diseño que dictan los manuales de la identidad marcaria de los productos UCI.

Para lograr un diseño simple y funcional, se propone un menú de navegación lateral que contenga las categorías de componentes simples, componentes avanzados, marcas, gráficas y paneles. Como elemento más común y distintivo entre los sistemas que tienen esta finalidad de negocio se encuentra un contenedor donde se muestre la vista previa. Esta característica es muy importante ya que es la que le brinda la posibilidad al usuario de lograr visualmente su diseño, poder seleccionar los elementos y ubicarlos en el lugar de la página de su preferencia. Estos elementos seleccionados deben se editados en cuanto a sus campos de textos y otros elementos dentro de cada componente.

Para lograr que perdure el diseño y pueda ser utilizado posteriormente, se dotaría al sistema de las funcionalidades de generar y exportar el código correspondiente. Esa característica garantiza la persistencia de los diseños para la posterior utilización por los proyectos de desarrollo de la universidad.

Para lograr cumplimentar la solución propuesta se definen las siguientes funcionalidades que darán valor al negocio:

- Gestionar componentes de selección.
- Gestionar componentes de formularios.
- Gestionar edición de componentes.
- Gestionar componentes asociados a la identidad marcaria.
- Generar código.
- Exportar código.

2.3. Fase de planificación

En la fase de planificación el cliente logra conformar las historias de usuario y a partir de estas, se crea un plan de iteraciones y de entregas que se van corrigiendo a partir de reuniones sistemáticas con el cliente y el equipo de trabajo. Esto le permite al equipo de desarrollo conocer las pretensiones del cliente y su vez familiarizarse con las herramientas y tecnologías que se utilizarán en el proyecto. El tiempo de duración de esta fase depende de la comunicación entre ambas partes.

2.3.1. Historias de usuario

Es unos de los artefactos base que genera la metodología XP, las cuales tienen el mismo objetivo que los casos de uso, pero son confeccionadas por el cliente. Estas se elaboran en base a las necesidades del sistema y está compuesta por descripciones cortas y en lenguaje natural. Proporcionan detalles sobre la estimación de riesgo y el tiempo estimado de su implementación (Izaurralde, 2013). A continuación, se muestra la descripción de las historias de usuarios del sistema. Las restantes están presentes en los anexos.

Tabla 2.1. Historia de usuario # 1

Historia de usuario	
Número: 1	**Nombre:** Adicionar componentes de selección
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario adicionar los componentes de selección disponibles en el sistema. • Botón simple • Tarjeta con descripciones y enlaces • Menú Dropdown	

Continúa en la próxima página

Tabla 2.1. Continuación de la página anterior

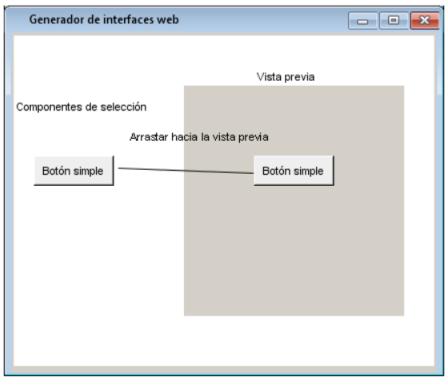
Observaciones: Se adicionará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado
Interfaz: 

Tabla 2.2. Historia de usuario # 2

Historia de usuario	
Número: 2	**Nombre:** Eliminar componentes de selección
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario eliminar los componentes de selección que había utilizado previamente. • Botón simple • Tarjeta con descripciones y enlaces • Menú Dropdown	
Observaciones: Se eliminará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado previamente	

Continúa en la próxima página

Tabla 2.2. Continuación de la página anterior

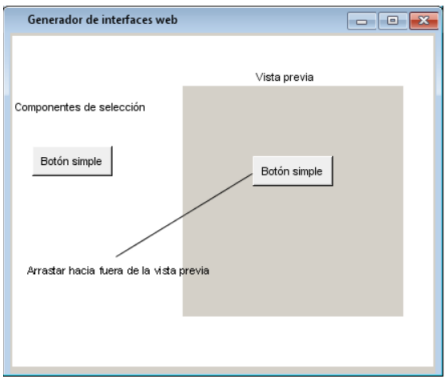
Interfaz: 

Tabla 2.3. Historia de usuario # 3

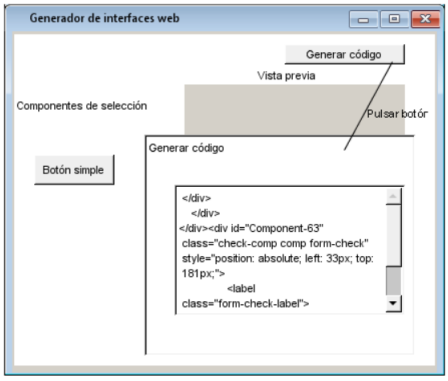
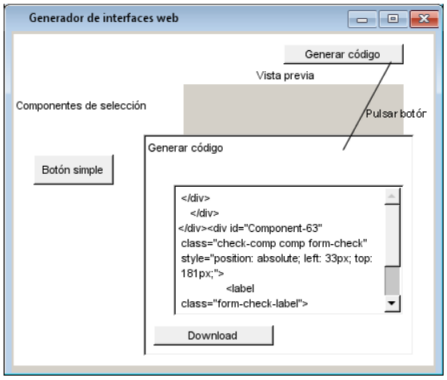
Historia de usuario	
Número: 3	**Nombre:** Generar código
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Frank López Acosta	
Descripción: Otorga la posibilidad al usuario de obtener el código fuente de la interfaz previamente generada.	
Observaciones: Se generará y se mostrará el código de la interfaz que el usuario diseñó previamente	
Interfaz: 	

Tabla 2.4. Historia de usuario # 4

Historia de usuario	
Número: 4	**Nombre:** Exportar código
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario exportar el código generado a partir de su diseño.	
Observaciones: Se exportará de manera comprimida al usuario el código fuente que se obtuvo a partir del diseño realizado previamente	
Interfaz: 	

2.4. Requisitos no funcionales

Los requisitos no funcionales son características propias del sistema. Definen condiciones que favorecen la calidad del producto en cuanto a restricciones, usabilidad, entre otros.

Para el desempeño del sistema propuesto se tuvieron en cuenta los siguientes requisitos no funcionales:

Requerimientos de apariencia o interfaz externa:

RNF 1: El sistema estará optimizado para todas las resoluciones por sus características de web adaptativa.

RNF 2: Los colores y las tipografías estarán ajustadas a la identidad marcaria. Serán definidas a partir de la marca que seleccione el usuario.

Requerimientos de usabilidad:

RNF 3: La aplicación debe ser concebida para ser utilizada por personas que tengan conocimientos básicos sobre informática.

Requerimientos de eficiencia:

RNF 4: El sistema debe ser capaz de responder con rapidez a las peticiones de los usuarios (1 segundo). Se tiene en cuenta este plazo de tiempo máximo estimado el cual debe ser mucho menor, pero si se encuentra en estos límites no debe afectar el funcionamiento del sistema.

Requerimientos de software:

RNF 5: Sistema Operativo: Se puede utilizar cualquier sistema operativo, ya que es multiplataforma.

RNF 6: Los dispositivos clientes que utilizarán la herramienta deben contar con navegadores web que soporten HTML5, CSS3 y JavaScript.

RNF 7: Uilizar un navegador web (Mozilla Firefox a partir de su versión 25, Google Chrome a partir de su versión 19.0.1084.52 u Opera a partir de su versión 6.00).

Requerimientos de Hardware:

RNF 8: Se necesitaría contar con un procesador a 1 GHZ o superior, 1 GB RAM o superior, lo cual garantizaría la fluidez necesaria para un óptimo funcionamiento del navegador web donde se utilice la plataforma.

Restricciones en el diseño y la implementación:

RNF 9: El análisis y diseño de la aplicación estará basado en la metodología de desarrollo XP.

RNF 10: Utilizar la herramienta CASE Visual Paradigm para la modelación del sistema mediante el lenguaje UML.

RNF 11: Desarrollar la solución utilizando el IDE NetBeans.

2.5. Planificación de entrega

En esta etapa el cliente establece las prioridades de las historias de usuario y a su vez los desarrolladores proponen la estimación del esfuerzo que tomaría la realización de estas. Se determina el cronograma conjuntamente con el cliente y se establece un plazo límite de tres meses para la entrega (Godoy y Kasiak, 2012).

2.5.1. Estimación de esfuerzo por historia de usuario

Tabla 2.5. Estimación de esfuerzo por historia de usuario

Iteración	Historias de usuario		Puntos estimados (semanas)
1	1	Adicionar componentes de selección	1
	2	Eliminar componentes de selección	1
	3	Adicionar componentes de formularios	1
	4	Eliminar componentes de formularios	1
2	5	Adicionar edición de componentes en la vista previa	1
	6	Eliminar edición de componentes en la vista previa	1
	7	Modificar edición de componentes en la vista previa	1
3	8	Adicionar componentes asociados a la identidad marcaria	2
	9	Eliminar componentes asociados a la identidad marcaria	2
4	10	Generar código	2
	11	Exportar código	2
Total			**15.0**

2.6. Iteraciones

En este apartado se describen las iteraciones sobre el sistema antes de su entrega. En la primera iteración se intenta establecer elementos básicos de la arquitectura del sistema para que sirva de base para el resto del proyecto. Esto no siempre es posible debido a que el cliente decide cuales historias de usuario se implementaran en cada iteración, ya que esto maximiza el valor del negocio. Luego de pasar la última iteración, el sistema contará con las funcionalidades necesarias para que entre en funcionamiento.

2.6.1. Plan de iteraciones

Iteración 1:

Tendrá como objetivo cumplir con las historias de usuario de la 1 a la 4. Estas son las referidas a las funcionalidades de gestionar los componentes web comunes tales como elementos de selección y de formularios. Constituye la fase fundamental en el desarrollo del generador de interfaces web, ya que conforma la

base de implementación de la propuesta de solución.

Iteración 2:

En esta iteración se implementarán las historias de usuario desde la 5 hasta la 7. Estas están basadas en la edición de los componentes web ya alojados en la vista previa. Los mismos podrán ser editados por el usuario.

Iteración 3:

La iteración 3 cumple con la implementación de las historias de usuario 8 y 9. Estas son las referidas a la gestión de los componentes web asociados a la identidad marcaria de los productos UCI. Esta iteración dotaría a la herramienta generadora de interfaces web de los elementos asociados a las marcas, que cumplan con los estándares requeridos en la política de diseño de la universidad para cada una de las mismas.

Iteración 4:

Tendrá como objetivo cumplir con las historias de usuario 10 y 11, que son las encargadas de implementar las historias de usuario referidas a las funcionalidades exportar y generar código. Finalizada la misma el sistema contaría con todas funcionalidades requeridas y ya estaría listo para aplicarle las pruebas necesarias por parte del cliente.

2.6.2. Plan de duración de las iteraciones

A continuación, se muestra en la siguiente tabla el plan de duración de iteraciones. El mismo establece la duración de las iteraciones, así como el orden que seguirá el proceso de desarrollo en cuanto a las historias de usuario. (Fuente: Elaboración Propia)

Tabla 2.6. Plan de duración de las iteraciones

Iteración	Historias de usuario		Duración (semanas)
1	1	Adicionar componentes de selección	4.0
	2	Eliminar componentes de selección	
	3	Adicionar componentes de formularios	
	4	Eliminar componentes de formularios	
2	5	Adicionar edición de componentes en la vista previa	3.0
	6	Eliminar edición de componentes en la vista previa	
	7	Modificar edición de componentes en la vista previa	
3	8	Adicionar componentes asociados a la identidad marcaria	4.0
	9	Eliminar componentes asociados a la identidad marcaria	
4	10	Generar código	4.0

Continúa en la próxima página

Tabla 2.6. Continuación de la página anterior

	11	Exportar código	
Total			**15.0**

2.6.3. Plan de entrega

A continuación, se muestra cómo ha quedado conformado el plan de entrega de acuerdo a las fechas propuestas para la realización de cada iteración, así como el versionado del sistema de acuerdo a su iteración.

Entregable	1ra Iteración	2da Iteración	3ra Iteración	4ta iteración
Generador de interfaces web para la identidad marcaria de los productos UCI.	0.2 25 enero/2018	0.4 20 febrero/2018	0.7 30 marzo/2018	1.0 30 mayo/2018

Figura 2.1. Plan de entrega (Fuente: Elaboración Propia).

2.7. Fase de diseño del sistema

A continuación, se define el patrón arquitectónico que se utilizará en la solución. Además, se describirá el funcionamiento de la misma, así como la descripción de los artefactos que exige la metodología para la representación del sistema.

2.7.1. Patrón arquitectónico del sistema

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Vignaga y Perovich, 2003).

El tipo de arquitectura basado en componentes tiene las siguientes características:

- Es un estilo de diseño para aplicaciones compuestas de componentes individuales.
- Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas.

- Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

Esta arquitectura será la utilizada en el generador de interfaces web ya que la misma brinda la posibilidad de implementar cada componente HTML por separado heredando sus atributos desde una clase constructora principal. El uso de esta práctica es fundamental en el desarrollo de futuros componentes que garantizarían proveer al sistema de una mayor disponibilidad de los mismos, brindándole más opciones al cliente.

Con respecto a la coordinación de los componentes en el generador de interfaces, los componentes se comunican uno con el otro por medio de interfaces. Cuando un componente ofrece servicios al resto del sistema, este adopta una interfaz proporcionada que especifica los servicios que otros componentes pueden utilizar, y cómo pueden hacerlo. El cliente no necesita saber sobre los funcionamientos internos del componente (su implementación) para hacer uso de la misma. Este principio resulta en componentes referidos como encapsulados.

Otra característica importante de los componentes es que son sustituibles, así que un componente puede sustituir a otro (en tiempo de diseño o tiempo de ejecución), si el componente sucesor cumple los requisitos del componente inicial (expresado por medio de las interfaces). Por lo tanto, los componentes pueden ser sustituidos por una versión actualizada o una alternativa sin romper el sistema en el cual operan.

2.7.2. Descripción del funcionamiento del sistema

En primer lugar, para lograr que el sistema cumpla sus funciones principales se deben describir las funcionalidades del mismo dando un punto de vista desde el diseño de la implementación. A continuación se describe la lógica que deberá seguir el generador de interfaces web.

Para mejorar la interacción para el usuario, los componentes web disponibles en el mismo deben ser seleccionados mediante la funcionalidad arrastrar y soltar de HTML5. En las interfaces gráficas de usuario, arrastrar y soltar es un gesto del dispositivo señalador en el que el usuario selecciona un objeto y lo arrastra a una ubicación diferente o a otro objeto de destino.

Arrastrar y soltar se considera un enfoque importante de construcción de sistemas enfocados a usuarios finales. A diferencia de los lenguajes de programación tradicionales basados en texto, los lenguajes de programación para el usuario final se basan en componentes visuales, como mosaicos o íconos, que son manipulados por los usuarios finales a través de interfaces de arrastrar y soltar.

La interacción soltar está basada en la vista previa que conforma el área de destino de esta funcionalidad. Para el diseño de los componentes web se deberá crear una clase base para el sistema en la cual estén contenidos todos los atributos de los componentes UCI y estos sean generados, además de contener clases con

responsabilidades asociadas a los eventos de los dispositivos de entrada, otras que contengan las interfaces para la selección y movimiento de los componentes web, así para elementos genéricos dentro del sistema.

La edición de los componentes se deberá hacer mediante una sencilla interacción del usuario como el clic derecho en el área del elemento en cuestión, lanzando una ventana emergente tipo modal, elemento de control gráfico subordinado a la ventana principal del sistema donde los usuarios deben interactuar con la ventana modal antes de que puedan regresar a la vista principal editando los campos correspondientes para cada elemento dentro del componente.

Luego se debe capturar todo el código que está dentro de la vista previa para lograr generarlo posteriormente y a su vez obtenerlo mediante un fichero. Esta funcionalidad es la más importante del sistema ya que brinda la posibilidad de hacer perdurable el código generado por el cliente.

En la siguiente figura, se muestra la representación arquitectónica del sistema:

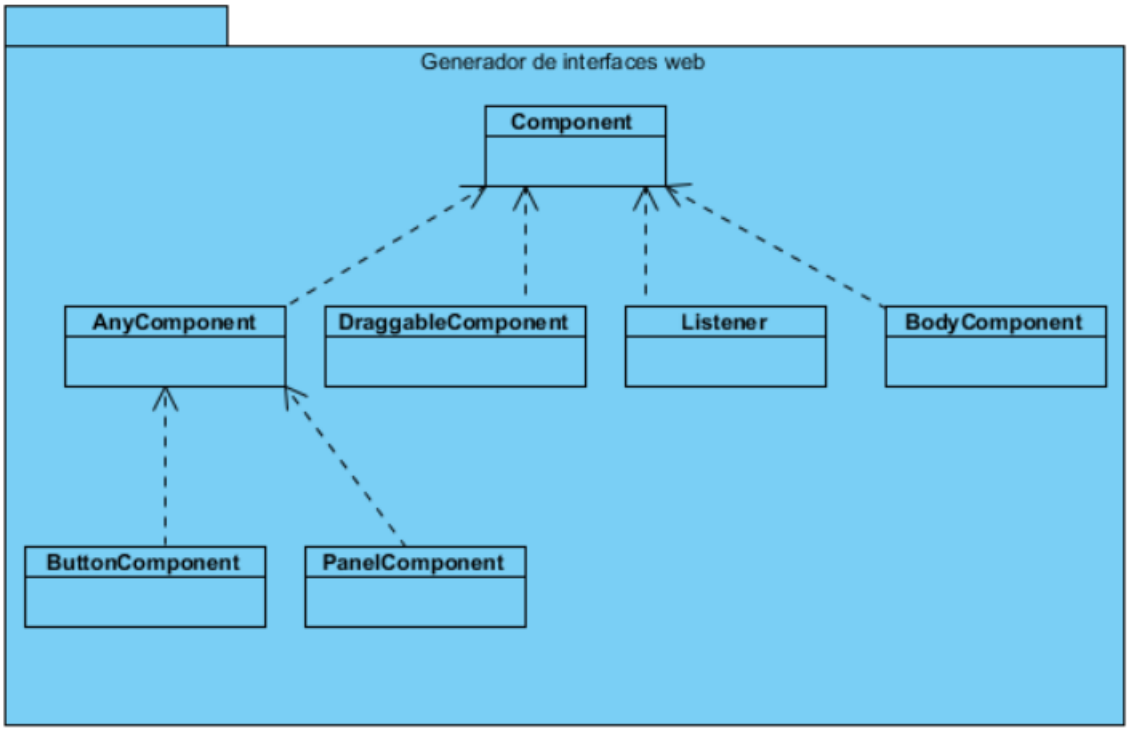


Figura 2.2. Representación de la arquitectura estructurada por clases(Fuente: Elaboración Propia).

A continuación, se describen las clases del sistema mediante las tarjetas Clase-Responsabilidad-Colaboración (CRC):

2.7.3. Tarjetas CRC

Las tarjetas CRC identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP (Gómez; Duarte y Guevara, 2014).

Tabla 2.7. Tarjeta CRC # 1

Tarjeta CRC	
Clase: Component	
Responsabilidad	**Colaboración**
• AddComponent(): Encargado de adicionar componentes HTML. • RemoveComponent(): Encargado de eliminar los componentes HTML. • Build(): Encargado de generar los componentes disponibles para su selección por el usuario.	AnyComponent BodyComponent Listener DraggableComponent

Tabla 2.8. Tarjeta CRC # 2

Tarjeta CRC	
Clase: AnyComponent	
Responsabilidad	**Colaboración**
• GenerateId(): Encargado de asignar nuevos identificadores a los componentes seleccionados por el usuario.	Component ButtonComponent

Tabla 2.9. Tarjeta CRC # 3

Tarjeta CRC	
Clase: BodyComponent	
Responsabilidad	**Colaboración**

Continúa en la próxima página

Tabla 2.9. Continuación de la página anterior

• NewInstance(): Encargada de dar acceso al objeto Body del HTML creando instancias de los componentes disponibles. • InitCompoment(): Encargada de inicializar los componentes dentro de la vista previa.	Component

Tabla 2.10. Tarjeta CRC # 4

Tarjeta CRC	
Clase: ButtonComponent	
Responsabilidad	**Colaboración**
• CloneComponent(): Encargada de clonar los atributos de los componentes disponibles para el usuario. • GetContextMenu(): Encargada de modificar los atributos a los componentes dentro de la vista previa.	Component AnyComponent

Tabla 2.11. Tarjeta CRC # 5

Tarjeta CRC	
Clase: DraggableComponent	
Responsabilidad	**Colaboración**
• DraggableListener(): Encargada de garantizar que los componentes web sean arrastrables. • DropListener(): Encargada de garantizar que los componentes web sean ubicados por el usuario en la posición requerida por el mismo dentro de la vista previa.	Component AnyComponent ButtonComponent Listener

Tabla 2.12. Tarjeta CRC # 6

Tarjeta CRC	
Clase: Listener	
Responsabilidad	**Colaboración**
• SimpleEventListener: Encargada de gestionar los eventos asociados al periférico de entrada(*Mouse*).	Component AnyComponent DraggableComponent ButtonComponent

Tabla 2.13. Tarjeta CRC # 7

Tarjeta CRC	
Clase: PanelComponent	
Responsabilidad	**Colaboración**
• MoreComponent(): Encargada de generar los componentes web asociados a la identidad marcaria.	Component AnyComponent

2.7.4. Patrones de diseño

Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable. El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y las ventajas e inconvenientes de su uso (Gamma, 2003). Para el diseño de las clases se utilizó el conjunto de patrones de diseño object-oriented design General Responsibility Assignment Software Patterns (GRASP, por sus siglas en inglés) y Gang of Four (GOF, por sus siglas en inglés), que intervienen en la asignación de responsabilidades, estructura y comportamiento respectivamente. A continuación, se indicarán aquellos que fueron aplicados.

Patrones GRASP

Se utiliza el **patrón Creador**, que tiene como objetivo guiar la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador encargado de producir el objeto en dependencia del contexto actual. Este patrón se puede evidenciar en la tarjetas CRC número 1 donde se describe la clase Component ya que es la clase encargada de crear las instancias correspondientes para cada uno de los componentes del sistema.

El **patrón Experto** representa un principio básico que suele utilizarse en el diseño orientado a objetos. Es la base de diseño donde el objeto de software realiza las operaciones que normalmente se aplican al

concepto que representa. Permitió la asignación de las responsabilidades necesarias a desarrollar a cuyas clases contienen la información para realizarlas. Este patrón se evidencia en la clase Component descrita en la tarjeta CRC número 1, es quien tiene acceso a todas las entidades necesarias y por ello a la información, por lo cual se le asigna la responsabilidad de generar todos los componentes web disponibles en el sistema.

El patrón **Bajo acoplamiento** es una medida de la fuerza en que una clase está conectada a otras, que la conoce y recurre a ellas. El objetivo de este patrón consiste en mantener un bajo nivel de dependencia de otros elementos, por lo que constituye un principio que debe estar presente en todas las decisiones de diseño con lo que se reduce el impacto de los cambios. Se pone de manifiesto en la clase BodyComponent descrita en la tarjeta CRC número 3.

El patrón **Alta cohesión** es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Se puede afirmar que cada una de las clases del sistema tiene alta cohesión, de manera que estas poseen la característica de tener las responsabilidades estrechamente relacionadas. Esta particularidad evita en cada caso, tener que realizar un trabajo enorme al garantizar un mejor diseño en ocasiones para el resultado global.

El patrón **Controlador** es en el cual se hace necesario conocer que un evento del sistema es una operación que se realiza en este, generada por un usuario externo. Un controlador, es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. En esta solución se encuentra ejemplo de la clase AnyComponent descrita en la tarjeta CRC número 2, que es la encargada de obtener todos los componentes web HTML para la generación de los mismos.

Patrones GOF

Se utilizó el patrón **Facade**, el cual tiene como propósito brindar una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar (Guerrero; Suárez y L. E. Gutiérrez, 2013). Se evidencia su uso en la clase Component, descrita en la tarjeta CRC número 1, la cual es una clase que además de ser la base del sistema es una interfaz unificada para las restantes clases que componen el mismo. El uso de este patrón garantiza que se oculten a los clientes los componentes del subsistema.

También se hace uso del patrón **Singleton**. Este garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella. El mismo, se pone de manifiesto en la clase BodyComponent descrita en la tarjeta CRC número 3, la cual es la encargada de dar acceso al cuerpo(*Body*) del HTML, que es el encargado de manejar los eventos asociados a la vista previa, solo puede generar una sola instancia del mismo. El empleo de este patrón trae consigo ventajas tales como el acceso controlado a una única instancia. Puesto que la clase singleton encapsula su única instancia, puede tener un control estricto sobre cómo y cuándo acceden a ella los clientes (Ramírez, 2004).

2.8. Conclusiones parciales

Atendiendo a los aspectos abordados en este capítulo referentes a las características funcionales de la solución a implementar, se arribaron a las siguientes conclusiones:

- La definición de las historias de usuario, brinda una guía descriptiva de cómo se comportara la implementación del sistema.
- Se identificaron como principales características funcionales de la solución a desarrollar, la generación de componentes de selección y componentes de interfaz de usuario basados en la identidad marcaria, la generación de código de las interfaces creadas, así como la exportación del código de dichas interfaces. Estas funcionalidades garantizan un generador de interfaces que satisfaga las necesidades de la investigación y a su vez facilite la generación de interfaces web asociadas a la identidad marcaria.
- La entrega de la versión final del sistema a partir de la estimación del tiempo para la implementación de las historias de usuario, quedó definida en tres meses y tres semanas. Este tiempo se ajusta a los tiempos de entrega propuestos para la investigación y cumple con las pautas que establece la metodología XP.
- Se generaron los artefactos correspondientes a las fases de Planificación y Diseño de la metodología de desarrollo XP. De esta forma quedan definidos los aspectos de arquitectura y diseño a tener en cuenta durante la fase de implementación de la solución.

CAPÍTULO 3

Implementación y prueba del sistema

3.1. Introducción

Este capítulo estará dedicado a las fases de implementación y prueba del sistema. Por otra parte se identifican y describen todas y cada una de las tareas de ingenierías llevadas a cabo por el equipo de desarrollo referentes a cada una de las historias de usuario. Se definen los prototipos de interfaz de usuario, así como los estándares de programación definidos. Por último, se muestra el proceso de pruebas de aceptación, con el objetivo de certificar que el sistema propuesto funcione de forma correcta.

3.2. Fase de implementación

Para la fase de implementación, la metodología utilizada plantea que las historias de usuario seleccionadas para ser implementadas en cada iteración, se van realizando durante el transcurso de la iteración a la cual pertenecen. Para ello se descomponen las historias de usuario en tareas de ingeniería. Su objetivo es definir cada una de las actividades que dan cumplimiento a las historias de usuario, de forma tal que se entienda lo que el sistema tiene que hacer y facilite su construcción. A continuación, se describen las tareas de ingeniería correspondientes a la propuesta de solución. Las restantes están presentes en los anexos.

Tabla 3.1. Tarea de ingeniería # 1

Tarea	
Número de tarea: 1	**Número de Historia de usuario:** 1
Nombre de la tarea: Adicionar componentes de selección	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 1 de febrero de 2018	**Fecha de fin:** 10 de febrero de 2018
Programador responsable: Frank López Acosta	

Continúa en la próxima página

Tabla 3.1. Continuación de la página anterior

Descripción: El usuario podrá seleccionar y adicionar los elementos que considere necesarios para elaborar la vista previa de su sitio web. Se debe otorgar la posibilidad al usuario de interactuar con los elementos a partir de arrastar y soltar los elementos de selección.

Tabla 3.2. Tarea de ingeniería # 2

Tarea	
Número de tarea: 2	**Número de Historia de usuario:** 2
Nombre de la tarea: Eliminar componentes de selección	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 11 de febrero de 2018	**Fecha de fin:** 20 de febrero de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario podrá seleccionar los elementos que forman parte de la vista previa de su sitio web y arrastrarlos fuera de la misma para su posterior eliminación.	

Tabla 3.3. Tarea de ingeniería # 3

Tarea	
Número de tarea: 3	**Número de Historia de usuario:** 4
Nombre de la tarea: Generar código	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 1 de mayo de 2018	**Fecha de fin:** 15 de mayo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario obtiene el código fuente en HTML de la interfaz generada por el mismo luego de finalizar su diseño.	

Tabla 3.4. Tarea de ingeniería # 4

Tarea	
Número de tarea: 4	**Número de Historia de usuario:** 5
Nombre de la tarea: Exportar código	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 16 de mayo de 2018	**Fecha de fin:** 30 de mayo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario obtiene el código fuente en HTML de la interfaz generada por el mismo luego de finalizar su diseño en un archivo comprimido.	

3.3. Prototipos de interfaz de usuario

En esta sección se muestran las interfaces que conformarían la solución. El mismo está sujeto a cambios y cuenta con los requisitos que se desean hasta el momento.

El sistema que se pretende desarrollar propone al usuario una interfaz sencilla en la cual estén incluidos elementos básicos para el diseño de una plantilla web enfocada en las reglas que rigen la identidad marcaria de los productos UCI. A partir de esto, el usuario es capaz de interactuar con el sistema a través de habilidades básicas, como arrastrar y soltar componentes web dentro de un contenedor para ir conformando las interfaces web pretendidas por el mismo.

En primer lugar, se propone una interfaz para la página principal donde el usuario pueda seleccionar una marca en específico, estas propuestas a través de un conjunto de tarjetas que contarán con el logo correspondiente y un enlace que lo direcciona hacia el generador de interfaces web. La figura siguiente muestra un prototipo de lo expuesto anteriormente:

Figura 3.1. Prototipo de página principal (Fuente: Elaboración propia).

El generador de interfaces web propone un diseño simple donde los usuarios puedan seleccionar los elementos a través de un menú desplegable lateral, donde se encuentran categorizados los mismos. Este a su vez, contaría con una vista previa, la cual ocuparía el mayor porcentaje de la interfaz de usuario y es donde el usuario pretendería realizar su diseño. A continuación, se muestra un prototipo del mismo:

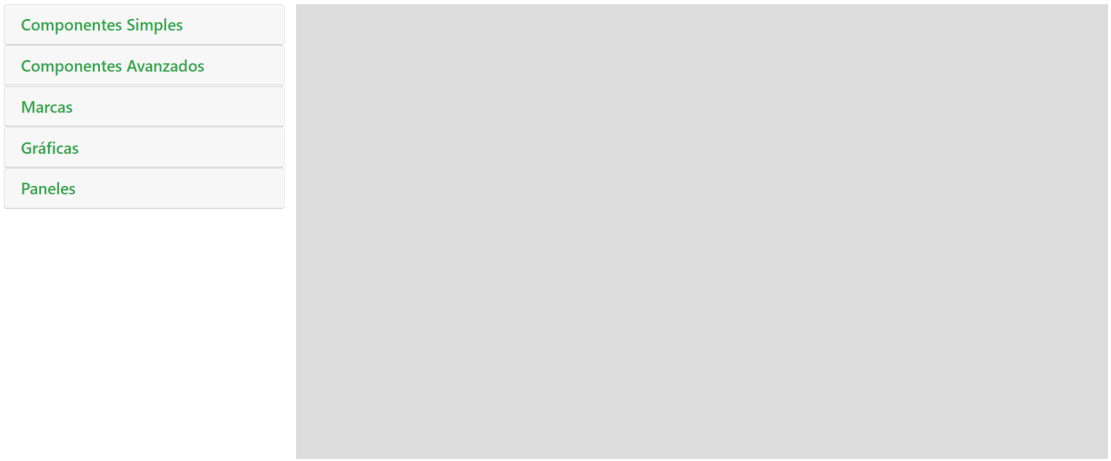


Figura 3.2. Prototipo del Generador de interfaces web (Fuente: Elaboración propia).

En su menú desplegable lateral, se encuentran las categorías asociadas a los componentes web. Ejemplo de ello, componentes simples, donde se encuentran englobados elementos comunes en el diseño web, tales como botones básicos, botones de radio, entre otros.

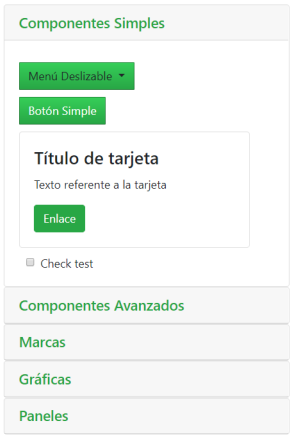


Figura 3.3. Prototipo de la categoría de componentes simples (Fuente: Elaboración propia).

Otra categoría sería la de los componentes avanzados, donde estarían predefinidos varios elementos que ya cumplen con la pautas de estilo y diseño propuestas por la universidad. Ejemplo de ello se muestra en la siguiente figura.

Figura 3.4. Prototipo de categoría de componentes avanzados (Fuente: Elaboración propia).

Además, contaría con la categoría referentes a las marcas, las gráficas de cada una de las marcas, así como paneles predefinidos, los cuales cumplen con las pautas que marca el manual de la identidad marcaria de los productos UCI. Estos prototipos estarán presentes en los anexos.

El sistema contará con componentes personalizables en algunos casos, luego de concluir su diseño tendrá la posibilidad de generar y exportar el código para su uso posterior.

3.4. Estándares de programación

Los estándares de programación son patrones que definen la estructura y apariencia del código para facilitar su lectura y entendimiento para otros desarrolladores. En el desarrollo del código se utilizaron varios que se proponen a continuación.

UpperCamelCase

Los nombres de las clases y las funciones adoptarán la notación UpperCamelCase y no se utilizará el guión bajo como delimitador entre palabras. A continuación se muestra su uso en el código fuente de la aplicación.

```
export class BodyComponent extends Component {
    constructor() {
        super('body');
        this.component = document.body;
    }
    static newInstance() {
        if (BodyComponent.instance == null) {
            BodyComponent.instance = new BodyComponent();
        }
        return BodyComponent.instance;
    }
```

Figura 3.5. Ejemplo de uso de UpperCamelCase (Fuente: Elaboración propia).

El **tamaño máximo de las líneas de código** debe ser de 80 a 100 caracteres aproximadamente, de manera tal que se garantice la completa visibilidad de las líneas de código sin necesidad de realizar desplazamiento horizontal. Ejemplo de ello en la siguiente figura.

```
static get(selector) {
    let elements = document.querySelectorAll(selector);
    let results = [];
    elements.forEach(function (element) {
        if (element[Component.ATTR_COMPONENT] instanceof Component) {
            results.push(element[Component.ATTR_COMPONENT]);
        }
        else {
            let temp = new AnyComponent(element);
            results.push(temp);
        }
    });
    return results;
}
```

Figura 3.6. Ejemplo limitación de líneas de código (Fuente: Elaboración propia).

Los **comentarios de implementación** se delimitarán entre los siguientes caracteres <!– –>. Este estándar es fundamental para un mejor entendimiento por parte de futuros desarrolladores.

```
<!--var draggeds = {};-->
<!--var dragged = null;-->

<!--var countId = 1;-->

<!--function enableDrag(element) {-->
    <!--element.attr('draggable', true);-->
    <!--element.attr('ondragstart', 'event.dataTransfer.setData("Data", this.id)');-->
```

Figura 3.7. Ejemplo de comentarios de implementación (Fuente: Elaboración propia).

Los nombres de las clases, métodos y variables serán escritos en idioma inglés. Con este estandar se garantiza una uniformidad general en el código fuente.

```
import { PanelComponent } from "./PanelComponent";
export class AnyComponent extends PanelComponent {
    constructor(element) {
        super();
        this.component = element;
        this.setComponent(this.component);
    }
}
```

Figura 3.8. Ejemplo del uso del idioma inglés (Fuente: Elaboración propia).

3.5. Fase de pruebas

Un desarrollo mediante la metodología de XP está compuesto por una serie de iteraciones cortas. Cada iteración concluye ejecutando un conjunto de pruebas de aceptación que permitan al cliente comprobar si está satisfecho con el resultado. Las pruebas son un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. Por esta razón se debe definir en el proceso de la ingeniería del software. Todo esto contribuye a elevar la calidad de los productos desarrollados y a la seguridad de los programadores a la hora de introducir cambios o modificaciones (Pressman, 2010).

3.5.1. Pruebas de aceptación

El propósito de las pruebas de aceptación es que los usuarios verifiquen que el sistema o los cambios satisfagan sus necesidades originales. El énfasis está en evaluar el sistema en un entorno de prueba controlado. Una de las misiones del cliente es escribir estas pruebas, realizarlas sobre el sistema y dar su aprobación al mismo. Al finalizar cada iteración se llevaron a cabo las respectivas pruebas previstas para las historias de

usuario que componen la misma (M. Mejías, 2014).

Las pruebas del sistema, tienen como objetivo verificar que el mismo cumple los requisitos establecidos por el usuario, por lo que también se puede incluir dentro de la categoría de pruebas de aceptación, según (J. Gutiérrez; Escalona; Mejías y Torres, 2006). En la investigación, se considera que las pruebas del sistema forman parte de las pruebas de aceptación. A continuación se muestran las pruebas de aceptación propuestas divididas por iteraciones, las restantes están presentes en los anexos.

Iteración 1

Tabla 3.5. Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Adicionar componentes de selección	
Descripción: Prueba para la funcionalidad agregar componentes de selección.	
Condiciones de ejecución: Los elementos deben estar disponibles para su elección por el usuario. El usuario debe seleccionar los elementos y arrastrarlos hasta la sección donde elabora su diseño.	
Pasos de ejecución: Arrastar los elementos hacia el área de destino donde se confeccionará el diseño.	
Resultados esperados: Los elementos seleccionados son arrastrables hacia el área de destino.	

Tabla 3.6. Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:** 2
Nombre: Eliminar componentes de selección	
Descripción: Prueba para la funcionalidad eliminar componentes de selección.	
Condiciones de ejecución: Los elementos deben estar ubicados en el área de destino.	
Pasos de ejecución: El usuario debe seleccionar los elementos y arrastrarlos hacia fuera del área de destino.	
Resultados esperados: Los elementos seleccionados son arrastrables hacia fuera del área de destino y se eliminan.	

Iteración 4

Tabla 3.7. Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU10_P10	**Historia de usuario:** 10
Nombre: Generar código	
Descripción: Prueba para la funcionalidad generar código.	
Condiciones de ejecución: El diseño propuesto por el usuario debe estar finalizado.	
Pasos de ejecución: Generar el código resultante del diseño del usuario.	
Resultados esperados: El usuario obtiene el código del diseño propuesto por el mismo.	

Tabla 3.8. Prueba de aceptación # 4

Caso de prueba de aceptación	
Código: HU11_P11	**Historia de usuario:** 11
Nombre: Exportar código	
Descripción: Prueba para la funcionalidad exportar código.	
Condiciones de ejecución: El diseño propuesto por el usuario debe estar finalizado.	
Pasos de ejecución: Exportar el código resultante del diseño del usuario.	
Resultados esperados: El usuario obtiene el código del diseño propuesto por el mismo en un archivo HTML.	

3.5.2. Resultados de pruebas de aceptación

Se realizaron un total de 11 pruebas de aceptación, de las cuales 3 de estas mostraron resultados no satisfactorios para el cliente. Esto representa un 27.7 porciento del total de pruebas realizadas. En la siguiente figura, se muestra los resultados de las pruebas de aceptación divididas por las iteraciones correspondientes:

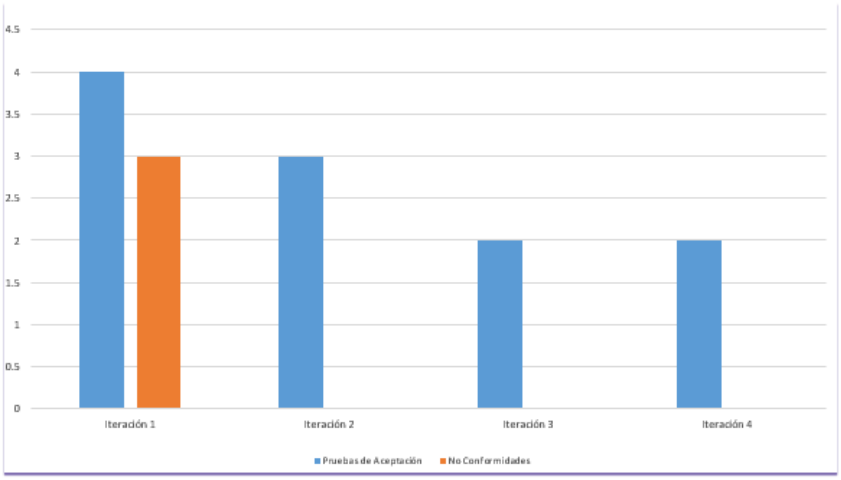


Figura 3.9. Resultados de pruebas de aceptación (Fuente: (Elaboración propia)).

Como se evidencia en la figura antes mostrada, el proceso de pruebas se realizó de forma iterativa en las 4 historias de usuarios que conforman la iteración 1. En ella se detectaron un total de 3 no conformidades por parte del cliente, representando un 75 % de las pruebas para la primera iteración. Las no conformidades encontradas, son debidas a que los componentes web seleccionados por el usuario no se mostraban correctamente en la vista previa. A partir de esta detección, se corrigieron las deficiencias utilizando pruebas de regresión, las cuales se centran en descubrir errores (*bugs*) o divergencias funcionales con respecto al comportamiento esperado del sistema.

En la iteración 2 no se encontraron no conformidades, cumpliendo las historias de usuario con los flujos de ejecución previstos en las 3 pruebas correspondientes a la misma, lográndose una satisfacción de un 100 % por parte del usuario.

Al realizar las pruebas de aceptación para la iteración 3, no se detectaron problemas en la ejecución de las mismas, cumpliendo con las exigencias demandadas por el usuario.

En la iteración 4 se realizaron las pruebas de aceptación referentes a las historias de usuario 10 y 11. En ellas no se detectaron problemas por el usuario. Esto representa que la solución desarrollada cumple con las exigencias del cliente, logrando así un 100 % de satisfacción por parte del mismo.

3.6. Conclusiones parciales

Atendiendo a los aspectos abordados en este capítulo, referentes a las fases de implementación y pruebas, se arribaron a las siguientes conclusiones:

- A partir de la generación de las tareas de ingeniería, se logró una descripción detallada de cada una

de las historias de usuarios, posibilitando un entendimiento favorable para los desarrolladores.

- Se establecieron estándares de código que facilitan la interpretación del mismo por otro equipo de desarrollo, garantizando el mantenimiento y actualización del sistema.
- Se obtuvieron resultados favorables al aplicar las pruebas de aceptación logrando una solución que satisfaga las necesidades del cliente.

Conclusiones

Luego de cumplir todas las tareas propuestas en la investigación se arriban a las siguientes conclusiones:

- La realización del estudio del arte, sobre los conceptos asociados a la investigación, marcaron las bases para el entendimiento de la solución.
- El estudio de sistemas homólogos, permitió definir funcionalidades elementales para el generador de interfaces web.
- El proceso de desarrollo que propone la metodología XP fue efectivo en el desarrollo del generador de plantillas web, lográndose la implementación del mismo y cumpliendo con todas las funcionalidades propuestas en el desarrollo de la investigación.
- Se validó el correcto funcionamiento del sistema a través de las pruebas de aceptación, siendo la misma fundamental, en la fase de pruebas que propone la metodología XP.
- La implementación del generador de plantillas web para obtener productos que apliquen a la identidad marcaria, se da cumplimiento al objetivo de la investigación y se logra facilitar el diseño de aplicaciones web desarrolladas en la UCI, ya que los componentes web disponibles cumplen con las exigencias del manual de identidad marcaria. Además, el sistema propuesto en la solución logra que los diseños sean perdurables en el tiempo y de fácil reutilización.

Recomendaciones

Para lograr que el generador de plantillas web cumpla con exigencias futuras por parte de los centros de producción se recomienda:

- Añadir categorías de componentes web para lograr un sistema donde estén disponibles la mayoría de los elementos que componen los proyectos de desarrollo de la universidad.
- Incorporar contenedores editables que permita al usuario estructurar los diseños previstos por los mismos.

Acrónimos

CMS Content Management Systems. 4–6
CRC Clase-Responsabilidad-Colaboración. 30, 33, 34
CSS Cascading Stylesheets. 9, 15, 18

GOF Gang of Four. 33, 34
GRASP object-oriented design General Responsibility Assignment Software Patterns. 33
GUI Graphical user interface. 6

HTML HyperText Markup Language. 5, 9, 18, 29, 32, 34

IDE Integrated Development Environment. 15, 18

UCI Universidad de las Ciencias Informáticas. 1, 2, 6, 19, 20, 22, 29, 42

XP Extreme Programming. 16–18, 23, 30, 34, 38, 42

Referencias bibliográficas

AXURE, RP, 2012. *Interactive wireframe software and mockup tool* (vid. pág. 10).

BAILÉN, María del Carmen Tolosa y BERNABEU, José Ramón García, 2011. "Google sitesçomo herramienta educativa:[Póster]. En: *"Google sitesçomo herramienta educativa:[Póster]. IX Jornades de xarxes d'investigació en docència universitària: Disseny de bones pràctiques docents en el context actual*, pág. 563 (vid. pág. 10).

BOLAGAY, Pillajo y ANDRÉS, Carlos, 2015. *Template genertor, software para la generación de plantillas web interactivas*. B.S. thesis. Universidad de las Fuerzas Armadas ESPE. Carrera de Ingeniería en Electrónica y Telecomunicaciones. (vid. pág. 7).

CUERDA, Xavier y MINGUILLÓN, Julià, 2004. Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto. *Mosaic*. Vol. 36 (vid. pág. 6).

EXPÓSITO, Erly Delgado, 2008. Metodologías de desarrollo de software tradicionales. *Revista de Arquitectura e Ingeniería*. Vol. 2, págs. 2008 (vid. pág. 14).

GAMMA, 2003. *Patrones de diseño*. Ed. por ADDISON-WESLEY (vid. pág. 33).

GARCÍA, Josep, 2014. ¿Qué es un creador de páginas web? (Vid. pág. 7).

GARCÍA, Onieva, 2009. Avanquest webeasy. *PC Actual. Computadora personal* (vid. págs. 9, 10).

GAUCHAT, Juan Diego, 2012. *El gran libro de HTML5, CSS3 y Javascript*. Marcombo (vid. pág. 18).

GODOY, Diego Alberto y KASIAK, Tamara, 2012. Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP. En: *Modelo dinámico de simulación para la gestión de proyectos de software desarrollados con XP. XVIII Congreso Argentino de Ciencias de la Computación* (vid. pág. 26).

GÓMEZ, Alveiro Rosado; DUARTE, Alexander Quintero y GUEVARA, Cesar Daniel Meneses, 2014. Desarrollo ágil de software aplicando programación extrema. *Revista Ingenio UFPSO*. Vol. 5, n.º 1, págs. 24-29 (vid. pág. 31).

GUERRERO, Carlos A; SUÁREZ, Johanna M y GUTIÉRREZ, Luz E, 2013. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*. Vol. 24, n.º 3, págs. 103-114 (vid. pág. 34).

GUTIERREZ, Demián, 2011. Métodos de desarrollo de software. *Recuperado el*. Vol. 22 (vid. pág. 14).

GUTIERREZ, Emmanuel, 2009. *JavaScript: Conceptos básicos y avanzados (bibliotecas Prototype y Script. aculo. us)*. Ediciones ENI (vid. pág. 18).

GUTIÉRREZ, JJ; ESCALONA, MJ; MEJÍAS, M y TORRES, J, 2006. Pruebas del Sistema en Programación Extrema. *Departamento de Lenguajes y Sistemas Informáticos. Universidad de Sevilla* (vid. pág. 41).

HERNÁNDEZ ORALLO, Enrique, 2000. El lenguaje unificado de modelado (UML). Vol. 20 (vid. pág. 17).

IZAURRALDE, María Paula, 2013. Caracterización de Especificación de Requerimientos en entornos ágiles: Historias de Usuario. *Trabajo de especialidad, Febrero* (vid. pág. 21).

LEDESMA, Rubén, 2008. Introducción al Bootstrap. Desarrollo de un ejemplo acompañado de software de aplicación. *Tutorials in Quantitative Methods for Psychology*. Vol. 4, n.º 2, págs. 51-60 (vid. pág. 17).

LETELIER, Patricio, 2006. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP) (vid. págs. 14, 16).

M. MEJÍAS, J. Torres, 2014. PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA (vid. pág. 41).

MARTÍNEZ, Alejandro y MARTÍNEZ, Raúl, 2014. Guía a rational unified process. *Escuela Politécnica Superior de Albacete–Universidad de Castilla la Mancha* (vid. pág. 15).

MONTAÑÉS, Pablo, 2011. Incomedia WebSiteX5 Evolution 9: crea tu web, bloq y negocio electrónico en cinco pasos. *PC Actual. Personal computer*. N.º 245, págs. 115-115 (vid. pág. 9).

ONIEVA GARCÍA, David, 2010. Magix Website Maker 4: diseñar, confeccionar y publicar tu propia página web nunca fue tan sencillo. *PC Actual. Personal computer*. N.º 230, págs. 119-119 (vid. pág. 10).

ORACLE, JDK, 2015. *7u80, Java SE, and NetBeans, Version 8.0. 2, Redwood City, CA, 2015* (vid. pág. 19).

PADILLA CARRASCAL, Jorge de Jesús, 2016. Creación y manejo de páginas web gratis a través de wix. com (vid. pág. 11).

PARADIGM, Visual, 2013. Visual paradigm for uml. *Visual Paradigm for UML-UML tool for software application development*, págs. 72 (vid. pág. 17).

PASCUAL, Francisco, 2008. Navegar en Internet: Adobe Dreamweaver CS3. *Alfaomega, 1era ED, México* (vid. pág. 9).

PÉREZ, Oiver Andrés, 2011. Cuatro enfoques metodológicos para el desarrollo de Software RUP–MSF–XP-SCRUM. *INVENTUM*. Vol. 6, n.º 10, págs. 64-78 (vid. pág. 15).

PRESSMAN, RS, 2010. *Ingeniería del Software UIn enfoque práctico, Septima edición ed*. McGraw-Hill (vid. pág. 40).

QUEVEDO ORTIZ, Daniel et al., 2013. Sistema para la generación automatizada de páginas web en el ámbito académico (vid. pág. 7).

RAMÍREZ, Alejandro, 2004. Introducción a los Patrones de Diseño. *Creative Commons. Agosto* (vid. pág. 34).

RAMOS MUÑOZ, Víctor et al., 2017. *Creación de lenguajes de programación optimizados con Truffle/-Graal*. B.S. thesis (vid. pág. 18).

RITTBERGER, Marc y BLEES, Ingo, 2009. Entorno de aprendizaje de la Web 2.0: Concepto, aplicación y evaluación. *En: eLearning Papers*. N.º 15 (vid. pág. 17).

ROMANÍ, CristobalCobo y KUKLINSKI, Hugo Pardo, 2008. *Planeta web 2.0: inteligencia colectiva. 2007* (vid. pág. 5).

SALA, Jesús Javier Rodríguez, 2003. *Introducción a la programación. Teoría y práctica.* (Vid. pág. 17).

SCHWABER, 2013. La guía definitiva para scrum: las reglas del juego. *Recuperado de: http: // www. scrumguides. org / docs / scrumguide / v1 / scrum-guide-us. pdf* (vid. pág. 16).

TINOCO GÓMEZ, Oscar; ROSALES LÓPEZ, Pedro Pablo y SALAS BACALLA, Julio, 2010. Criterios de selección de metodologías de desarrollo de software. *Industrial Data*. Vol. 13, n.º 2 (vid. pág. 15).

TRAMULLAS, J y GARRIDO, P, 2006. Los sistemas de gestión de contenidos. *Tendencias en documentación digital. Gijón: Trea*, págs. 135-161 (vid. pág. 6).

TURNER, Michael, 2006. *Microsoft® solutions framework essentials: building successful technology solutions*. Microsoft Press (vid. pág. 15).

VALLÉS, Juan Enrique Gonzálvez, 2011. *La Web 2, 0 Y 3, 0 en Su Relación Con El Eees*. Editorial Visión Libros (vid. pág. 6).

VAN DER HENST, Christian, 2005. Qué es la Web 2.0. *Recuperado el*. Vol. 15 (vid. pág. 5).

VIGNAGA, Andrés y PEROVICH, Daniel, 2003. Enfoque metodológico para el desarrollo basado en componentes. *Artículo Extraído el*. Vol. 21 (vid. pág. 28).

Generado con LaTeX: 27 de junio de 2018: 12:11am

Apéndices

APÉNDICE A

Anexos 1

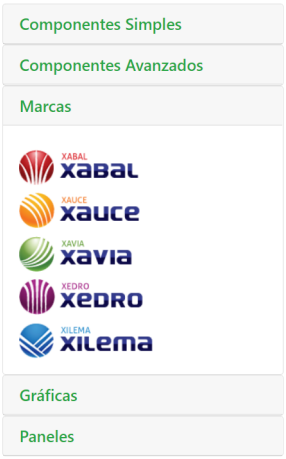


Figura A.1. Prototipo de categoría de marcas (Fuente: Elaboración propia).

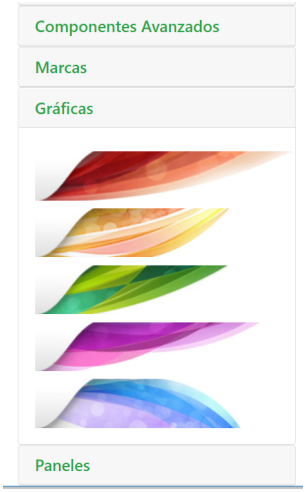


Figura A.2. Prototipo de categoría gráficas (Fuente: Elaboración propia).

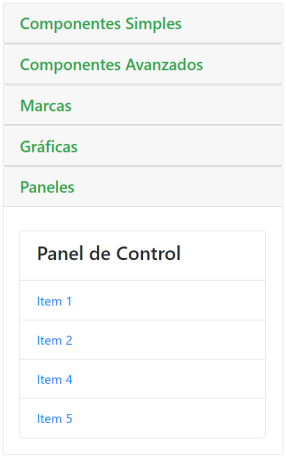


Figura A.3. Prototipo de categoría paneles (Fuente: Elaboración propia).

A.0.1. Resultados obtenidos

Figura A.4. Resultados de la vista principal (Fuente: Elaboración propia).

Figura A.5. Resultados de una interfaz XAVIA (Fuente: Elaboración propia).

Figura A.6. Resultados de una interfaz XABAL (Fuente: Elaboración propia).

Figura A.7. Resultados de una interfaz XAUCE (Fuente: Elaboración propia).

Figura A.8. Resultados de una interfaz XEDRO (Fuente: Elaboración propia).

Figura A.9. Resultados de una interfaz XILEMA (Fuente: Elaboración propia).

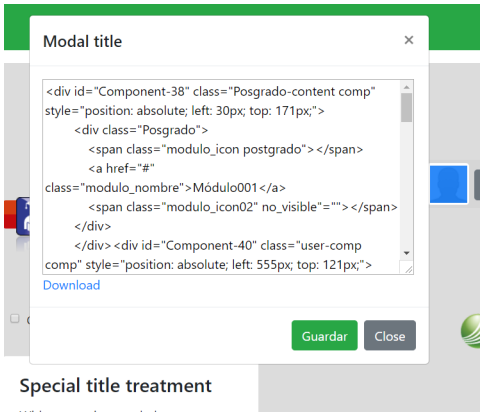


Figura A.10. Vista de la funcionalidad generar código (Fuente: Elaboración propia).

APÉNDICE B

Anexo 2

B.1. Historias de usuario

Tabla B.1. Historia de usuario # 5

Historia de usuario	
Número: 5	**Nombre:** Adicionar componentes de formularios
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario adicionar a los componentes de formularios disponibles en el sistema. • CheckTest • Ejemplo de radio • Select • File • Etiqueta • Input	
Observaciones: Se adicionará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado	

Continúa en la próxima página

Tabla B.1. Continuación de la página anterior

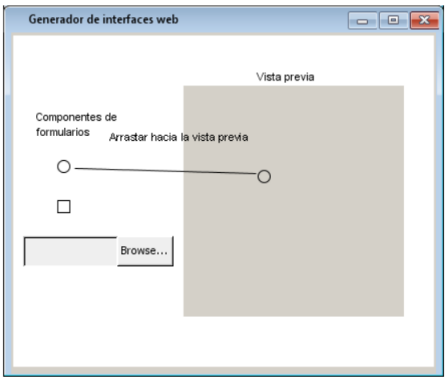
Interfaz: 

Tabla B.2. Historia de usuario # 6

Historia de usuario	
Número: 6	**Nombre:** Eliminar componentes de formularios
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario eliminar los componentes de formularios que había utilizado previamente. • CheckTest • Ejemplo de radio • Select • File • Etiqueta • Input	
Observaciones: Se eliminará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado previamente	

Continúa en la próxima página

Tabla B.2. Continuación de la página anterior

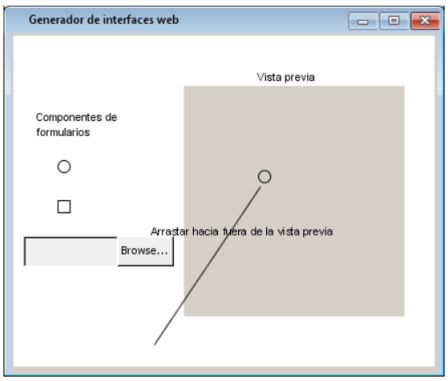
Interfaz:


Tabla B.3. Historia de usuario # 7

Historia de usuario	
Número: 7	**Nombre:** Adicionar edición de componentes
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario editar los componentes web seleccionados, editando sus campos y agregandos características a fines con cada componente.	
Observaciones: Se adicionará la edición propuesta por el usuario	
Interfaz:	

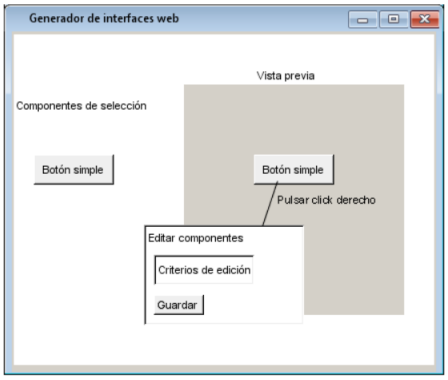


Tabla B.4. Historia de usuario # 8

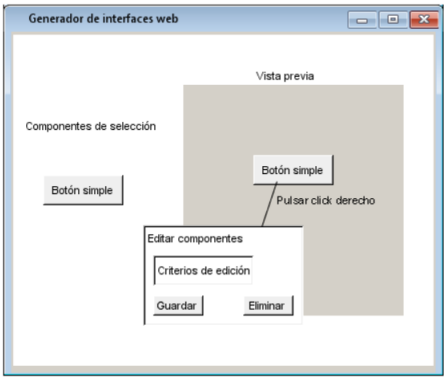
Historia de usuario	
Número: 8	**Nombre:** Eliminar edición de componentes
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario eliminar la edición de los componentes web seleccionados.	
Observaciones: Se eliminará la edición propuesta por el usuario	
Interfaz: 	

Tabla B.5. Historia de usuario # 9

Historia de usuario	
Número: 9	**Nombre:** Modificar edición de componentes
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario modificar la edición de los componentes web seleccionados, editando sus campos y agregando características a fines con cada componente.	
Observaciones: Se modificará la edición propuesta por el usuario	

Continúa en la próxima página

Tabla B.5. Continuación de la página anterior

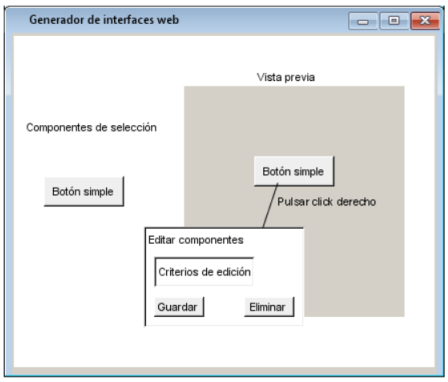
Interfaz: 

Tabla B.6. Historia de usuario # 10

Historia de usuario	
Número: 10	**Nombre:** Adicionar componentes asociados a la identidad marcaria
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario adicionar los componentes asociados a la identidad marcaria en el sistema. • Gráficas • Paneles • Componentes avanzados	
Observaciones: Se adicionará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado	

Continúa en la próxima página

Tabla B.6. Continuación de la página anterior

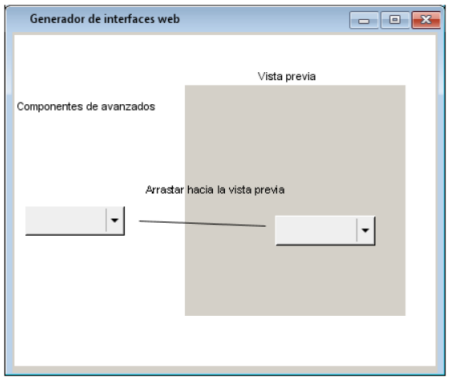
Interfaz: 

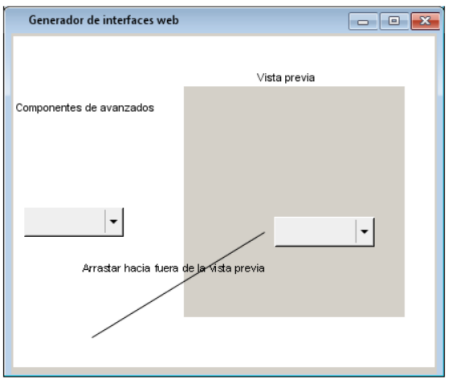
Tabla B.7. Historia de usuario # 11

Historia de usuario	
Número: 11	**Nombre:** Eliminar componentes asociados a la identidad marcaria
Usuario: Cliente	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 2	**Iteración asignada:** 3
Programador responsable: Frank López Acosta	
Descripción: Permite al usuario eliminar los componentes asociados a la identidad marcaria en el sistema. • Gráficas • Paneles • Componentes avanzados	
Observaciones: Se eliminará el componente seleccionado por el usuario en el contenedor donde haya sido arrastrado	

Continúa en la próxima página

Tabla B.7. Continuación de la página anterior

Interfaz:



B.2. Tareas de ingeniería

Tabla B.8. Tarea de ingeniería # 5

Tarea	
Número de tarea: 5	**Número de Historia de usuario:** 3
Nombre de la tarea: Adicionar componentes de formularios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 20 de febrero de 2018	**Fecha de fin:** 28 de febrero de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario prodrá seleccionar y adicionar los elementos de formularios que considere necesarios para elaborar la vista previa de su sitio web. Se debe otorgar la posibilidad al usuario de interactuar con los elementos a partir de arrastar y soltar los elementos de formularios.	

Tabla B.9. Tarea de ingeniería # 6

Tarea	
Número de tarea: 6	**Número de Historia de usuario:** 4
Nombre de la tarea: Eliminar componentes de formularios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 1 de marzo de 2018	**Fecha de fin:** 7 de marzo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario podrá seleccionar los elementos que forman parte de la vista previa de su sitio web y arrastrarlos fuera de la misma para su posterior eliminación.	

Tabla B.10. Tarea de ingeniería # 7

Tarea	
Número de tarea: 7	**Número de Historia de usuario:** 5
Nombre de la tarea: Adicionar edición de componentes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 8 de marzo de 2018	**Fecha de fin:** 10 de marzo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario adiciona un criterio de edición para cualquier componente web seleccionado previamente por el mismo a través de una ventana modal.	

Tabla B.11. Tarea de ingeniería # 8

Tarea	
Número de tarea: 8	**Número de Historia de usuario:** 6
Nombre de la tarea: Eliminar edición de componentes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 11 de marzo de 2018	**Fecha de fin:** 20 de marzo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario elimina un criterio de edición para cualquier componente web seleccionado previamente por el mismo	

Tabla B.12. Tarea de ingeniería # 9

Tarea	
Número de tarea: 9	**Número de Historia de usuario:** 7
Nombre de la tarea: Modificar edición de componentes	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha de inicio: 21 de marzo de 2018	**Fecha de fin:** 31 de marzo de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario modifica un criterio de edición para cualquier componente web seleccionado previamente por el mismo	

Tabla B.13. Tarea de ingeniería # 10

Tarea	
Número de tarea: 10	**Número de Historia de usuario:** 8
Nombre de la tarea: Adicionar componentes asociados a la identidad marcaria	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 1 de abril de 2018	**Fecha de fin:** 10 de abril de 2018
Programador responsable: Frank López Acosta	

Continúa en la próxima página

Tabla B.13. Continuación de la página anterior

Descripción: El usuario adiciona componentes web asociados a la identidad marcaria de los productos UCI.

Tabla B.14. Tarea de ingeniería # 11

Tarea	
Número de tarea: 11	**Número de Historia de usuario:** 9
Nombre de la tarea: Eliminar componentes asociados a la identidad marcaria	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha de inicio: 11 de abril de 2018	**Fecha de fin:** 20 de abril de 2018
Programador responsable: Frank López Acosta	
Descripción: El usuario elimina componentes web asociados a la identidad marcaria de los productos UCI.	

B.3. Pruebas de aceptación

Continuación de las pruebas de aceptación para la **iteración 1**:

Tabla B.15. Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 4
Nombre: Adicionar componentes de formularios	
Descripción: Prueba para la funcionalidad agregar componentes de formularios	
Condiciones de ejecución: Los elementos deben estar disponibles para su elección por el usuario. El usuario debe seleccionar los elementos y arrastrarlos hasta la sección donde elabora su diseño.	
Pasos de ejecución: Arrastar los elementos hacia el área de destino donde se confeccionará el diseño.	
Resultados esperados: Los elementos seleccionados son arrastrables hacia el área de destino.	

Tabla B.16. Prueba de aceptación # 6

Caso de prueba de aceptación	
Código: HU4_P4	**Historia de usuario:** 4
Nombre: Eliminar componentes de formularios	
Descripción: Prueba para la funcionalidad eliminar componentes de formularios	

Continúa en la próxima página

Tabla B.16. Continuación de la página anterior

Condiciones de ejecución: Los elementos deben estar ubicados en el área de destino
Pasos de ejecución: El usuario debe seleccionar los elementos y arrastrarlos hacia fuera del área de destino.
Resultados esperados: Los elementos seleccionados son arrastrables hacia fuera del área de destino y se eliminan.

Iteración 2:

Tabla B.17. Prueba de aceptación # 7

Caso de prueba de aceptación	
Código: HU5_P5	**Historia de usuario:** 5
Nombre: Agregar edición de compoonentes	
Descripción: Prueba para la funcionalidad adicionar edición de componentes	
Condiciones de ejecución: Los elementos deben estar en el área de vista previa. El usuario debe pulsar click derecho sobre el componente web a editar.	
Pasos de ejecución: Luego de pulsar click derecho sobre el componente se debe mostrar una ventana de tipo modal con los criterios de edición y pulsar en la opción guardar.	
Resultados esperados: Los componentes web son editados y sus cambios están guardados	

Tabla B.18. Prueba de aceptación # 8

Caso de prueba de aceptación	
Código: HU4_P4	**Historia de usuario:** 6
Nombre: Eliminar edición de componentes	
Descripción: Prueba para la funcionalidad eliminar edición de componentes	
Condiciones de ejecución: Los elementos deben estar ya editados en el área de vista previa.	
Pasos de ejecución: El usuario debe arrastrar el componente editado en la vista previa hacia fuera del contenedor para su posterior eliminación.	
Resultados esperados: Los componentes web editados son eliminados	

Tabla B.19. Prueba de aceptación # 9

Caso de prueba de aceptación	
Código: HU7_P7	**Historia de usuario:** 7
Nombre: Modificar edición de componentes	
Descripción: Prueba para la funcionalidad modificar edición de componentes	
Condiciones de ejecución: Los elementos deben estar en el área de vista previa ya editados. El usuario debe pulsar click derecho sobre el componente web a editar.	
Pasos de ejecución: Luego de pulsar click derecho sobre el componente se debe mostrar una ventana de tipo modal con los criterios de edición y pulsar en la opción guardar	
Resultados esperados: Los componentes web son editados y sus cambios están guardados	

Iteración 3

Tabla B.20. Prueba de aceptación # 10

Caso de prueba de aceptación	
Código: HU8_P8	**Historia de usuario:** 8
Nombre: Agregar componentes asociados a la identidad marcaria	
Descripción: Prueba para la funcionalidad agregar componentes asociados a la identidad marcaria	
Condiciones de ejecución: Los componentes deben estar disponibles para su elección por el usuario. El usuario debe seleccionar los componentes y arrastrarlos hasta la sección de vista previa donde elabora su diseño.	
Pasos de ejecución: Arrastar los elementos hacia el área de destino donde se confeccionará el diseño.	
Resultados esperados: Los elementos seleccionados son arrastrables hacia el área de destino.	

Tabla B.21. Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU9_P9	**Historia de usuario:** 9
Nombre: Eliminar componentes asociados a la identidad marcaria	
Descripción: Prueba para la funcionalidad eliminar componentes asociados a la identidad marcaria	
Condiciones de ejecución: Los componentes deben estar disponibles en el área de vista previa.	

Continúa en la próxima página

Tabla B.21. Continuación de la página anterior

Pasos de ejecución: Arrastar los elementos hacia fuera del área de destino donde se eliminará posteriormente.
Resultados esperados: Los elementos seleccionados son arrastrables hacia fuera del área de destino y son eliminados.