Facultad 4

Componentes personalizables para el marco de trabajo de desarrollo de multimedia con tecnologías libres

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Autores:

Susana Becerra Rodriguez

Daniel Díaz León

Tutores:

Ing. Lizandra Hernández Hernández

Ing. Alejandro Rodríguez de la Cruz

La Habana, 21 junio de 2016

"Año 58 de la Revolución"

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estime pertinente con el mismo.

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

Autor(es):

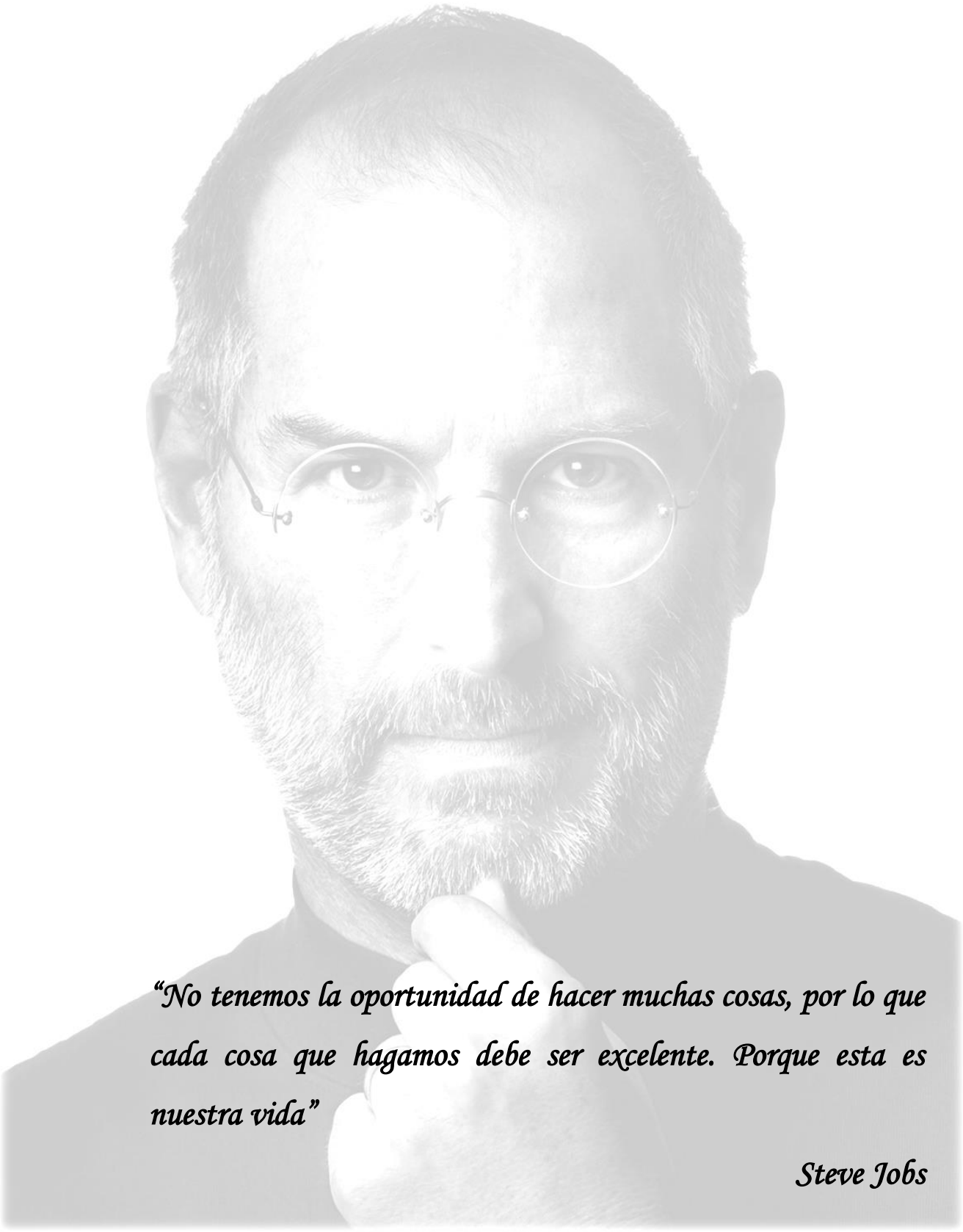
_____ _____

Susana Becerra Rodriguez Daniel Díaz León

Tutor(es):

_____ _____

Ing. Lizandra Hernández Hernández Ing. Alejandro Rodríguez de la Cruz



"No tenemos la oportunidad de hacer muchas cosas, por lo que cada cosa que hagamos debe ser excelente. Porque esta es nuestra vida"

Steve Jobs

De Susana Becerra Rodriguez

A mis padres: Carmen y Jacinto, porque sin ellos nada de esto hubiera sido posible si no fuera por su dedicación e interés por mi superación y por todo el sacrificio que han tenido que hacer para que hoy finalmente pudiera ser Ingeniera.

A mi hermano: Didiet por ser mi ejemplo a seguir.

A mi abuelo: Arcenio por todo el amor que me brinda y por estar siempre pendiente de mí.

A la memoria de mis otros abuelos que aunque la vida no les dio la oportunidad de estar presentes hoy aquí sé que estarían orgullosos de mí por las cosas que he podido lograr.

A mi familia en general, por su amor y dedicación, por su apoyo incondicional durante el transcurso de mi vida y por estar siempre presente de una forma u otra en los momentos importantes.

De Daniel Díaz León

A mi mamá Estrella por ser la mejor del mundo, por darme su amor y apoyo incondicional desde el día en que vine al mundo y guiarme por el camino correcto.

A mis hermanos Yadniel y Yerenis por ser ejemplos a seguir para mí y por hacerme saber que puedo contar con ellos en cualquier momento.

A toda mi familia por ser el motor impulsor de esta carrera, quiero regalarles este momento y agradecerles por tanto amor y dedicación.

De Susana Becerra Rodriguez

A mi mamá, agradecerle por su amor infinito y ser mi guía, por esos maravillosos consejos que solo ella brinda, por creer siempre en mí y darme ánimos de continuar luchando por lo quiero, por todo los sacrificios que ha tenido que hacer para lograr que yo pudiera hacer realidad mis mayores sueños en la vida, gracias mamita por existir y formar parte de mí vida.

A mi papá, le agradezco por su amor y dedicación hacia mí, por su apoyo incondicional y por los sacrificios que ha hecho para que yo pudiera salir adelante, papi te quiero.

A mi hermano por estar pendiente de mí y por estar presente cuando más lo necesito.

A mi tío Jesús y mi abuelo Arcenio por estar siempre pendientes de mí, por su amor y apoyo incondicional.

A Elena, Yeney, Rosmelys y María por estar siempre pendientes de mí, por su apoyo incondicional y de una forma u otra siempre están presentes en los momentos más importantes de mi vida.

A toda mi familia por su amor, confianza y apoyo.

A nuestros tutores Lizandra y Alejandro por su paciencia y porque siempre nos apoyaron y ayudaron.

A mi compañero de tesis por su paciencia y dedicación al trabajo, lo que demuestra que todo es posible cuando nos lo proponemos.

A mi amigo Daimel que gracias a esta universidad pude conocerlo, el cual en estos momentos lo considero como un hermano, te agradezco por toda la confianza, apoyo y ayuda en todo momento, especialmente cuando el camino se hacía difícil .

A las amistades que pude conocer en esta universidad durante estos 5 años, les agradezco por su compañía en algún momento de mi vida, en especial a mis compañeras de apartamento (Yaritza, Rosmelys, Yinelis y Cleidis) porque de una forma u otra formaron parte de mi familia.

A mis compañeros de aula gracias por pasar estos años juntos.

A todas las personas que de una forma u otra han tenido que ver con mi formación como profesional.

De Daniel Díaz León

A mi madre por ser lo más grande que tengo y por su eterna preocupación por mí, por hacer de mí una mejor persona y por su ejemplo.

A mi familia por apoyarme siempre y compartir buenos momentos en la vida.

A mis tíos Albe, Junca y Papo por ser como un padre para mí y guiarme por el buen camino.

A mis amigos Danis, Geovelsi por darme su apoyo y amistad durante estos años.

A mi compañera de tesis por su dedicación al trabajo, ha sido muestra de lo que se puede alcanzar cuando nos proponemos una meta.

A mis abuelos por todo el cariño y su comprensión.

A los tutores, por la confianza y por toda la ayuda que nos han dado.

A los amigos del IPI, a pesar de que a muchos nos tocó separarnos y ya casi ni nos vemos, aún conservo grandes amistades de ahí.

A los profesores que me han acompañado y guiado en estos años de universidad, gracias a todos por compartir conmigo más que sus conocimientos y ayudarme a ser mejor.

A mis compañeros de aula por su apoyo y amistad.

A todas mis amistades que siempre se preocuparon por mí a pesar de la distancia.

A todo el que me faltó por mencionar y que me ayudó durante mis estudios, son muchos los que han formado parte de mí durante estos 5 años.

Resumen

El proyecto Marco de trabajo de desarrollo de multimedia con tecnologías libres posee un conjunto de componentes básicos que le permiten al usuario crear una multimedia. En la actualidad los componentes poseen plantillas de diseño predeterminadas, por lo que no se ajusta a los requerimientos del usuario. Además, poseen baja adaptabilidad en pantalla para los dispositivos móviles. Por esta razón la presente investigación tiene como objetivo desarrollar componentes personalizables para el Marco de trabajo de desarrollo de multimedia con tecnologías libres. Los mismos serán integrados a un entorno de desarrollo para la confección de multimedia. Para guiar el desarrollo de los componentes se utilizó Proceso Unificado Ágil (AUP), generándose los artefactos fundamentales que propone la metodología para cada etapa de trabajo. Se utilizó HTML 5, CSS 3 y JavaScript como lenguajes de programación; como herramientas y tecnologías se emplearon Visual Paradigm 8.0 para el modelado UML, AngularJS 1.4.9, Bootstrap 3.1.1, Nodejs 0.12 y WebStorm 11.0. Como resultado, los usuarios podrán contar con plantillas de diseño cuyos elementos son configurables y se ajusten a sus necesidades, además de facilitar la confección de todo tipo de multimedia. Los componentes personalizables son adaptables a los diferentes dispositivos móviles y pueden ser reutilizados en productos que dentro de su arquitectura tecnológica empleen AngularJS. Se realizaron las pruebas de calidad lo que permitió detectar los errores en los componentes. Las no conformidades encontradas fueron corregidas, por tanto los componentes cumplen con la calidad y las funcionalidades establecidas por el cliente.

Palabras clave: componente personalizable, multimedia

Índice

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) son utilizadas en distintas esferas de la sociedad como son: salud, deporte, economía, defensa y educación. Cuba es uno de los países que a pesar de ser subdesarrollado, logró la nacionalización de la educación, siendo este uno de sus mayores logros. En la actualidad ha sido necesario vincular la educación con las tecnologías, debido al gran avance tecnológico presente en el mundo. Como consecuencia, a los profesores les ha sido necesario utilizar herramientas informáticas que sirvan de apoyo a las clases, surgiendo así el software educativo.

El software educativo se caracteriza por ser interactivo, debido al empleo de multimedia, fotografías, juegos, diccionarios especializados, ejercicios y cuestionarios con el fin de evaluar a los estudiantes. En la Universidad de las Ciencias Informáticas (UCI) se crean productos informáticos de este tipo, con el propósito de ayudar en el aprendizaje de los estudiantes. En la Universidad existen varios centros de producción, como el Centro de Tecnologías para la Formación (FORTES), perteneciente a la facultad 4. El mismo tiene como misión desarrollar tecnologías que permitan ofrecer servicios y productos. Con el fin de brindar soluciones de formación aplicando las TIC, a todo tipo de instituciones con diferentes modelos de formación y condiciones tecnológicas. Entre los proyectos que pertenecen a FORTES se encuentra el Marco de trabajo de desarrollo de multimedia con tecnologías libres.

El Marco agrupa un conjunto de componentes que facilitan la confección de una multimedia, disminuyen el tiempo de desarrollo, son multiplataforma y adaptables a los dispositivos móviles. También pueden ser reutilizados en los productos que dentro su arquitectura tecnológica empleen AngularJS. En la actualidad estos componentes sólo poseen plantillas de diseño predeterminadas. Esto trae consigo que en ocasiones no se ajusten a los requerimientos del usuario. Así como también posee baja adaptabilidad en pantalla para los diferentes dispositivos. Además, impide que el usuario pueda interactuar con el formato de texto de la aplicación.

Descrita la situación problémica se plantea como **problema a resolver**: ¿Cómo contribuir al proceso de creación de multimedia dentro del marco de trabajo de desarrollo de multimedia con tecnologías libres?

Se establece como **objeto de estudio** el proceso de desarrollo de multimedia con tecnologías libres.

Se define como **campo de acción** de la investigación los componentes personalizables para la creación de multimedia con tecnologías libres.

El **objetivo general** de la investigación es desarrollar componentes personalizables para el marco de trabajo de desarrollo de multimedia con tecnologías libres.

Los **objetivos específicos** a cumplir son:

- Elaborar el marco teórico de la investigación a partir del estudio del estado del arte sobre el tema.
- Implementar los componentes personalizables para el Marco de trabajo de desarrollo de multimedia.
- Realizar las pruebas necesarias para garantizar la funcionalidad y calidad de los componentes.

Las **tareas a cumplir** son:

- Revisión bibliográfica relacionada con el problema de la investigación.
- Elaboración del diseño teórico de la investigación.
- Diseño de los componentes personalizables a partir del análisis de los requerimientos.
- Implementación de los componentes personalizables.
- Realización de las pruebas con el fin de garantizar la funcionalidad y calidad de los componentes.

La presente investigación está sustentada por la **hipótesis**: Si se incorporan los componentes personalizables al marco de trabajo de desarrollo de multimedia, se contribuirá con el proceso de creación de software multimedia con tecnologías libres.

Los **resultados** a obtener de esta investigación serían los componentes personalizables para el marco de trabajo de desarrollo de multimedia con tecnologías libres. Permitiéndole al usuario contar con una plantilla cuyos elementos sean configurables, brindando la posibilidad de crear un producto que se ajuste a sus requerimientos.

El método científico es una serie de etapas a recorrer para obtener un conocimiento válido. Es un proceso para establecer relaciones entre los hechos y enunciar leyes que expliquen los fenómenos físicos del mundo. Con el fin de realizar una correcta investigación se seleccionarán los métodos científicos, los cuales se adaptan al objetivo general y a las tareas de investigación. A continuación, se citan los métodos teóricos y empíricos seleccionados:

Métodos teóricos

Histórico- Lógico: se empleó en el estudio de la evolución del proceso de desarrollo de multimedia con tecnologías libres.

Analítico- Sintético: se utilizó para realizar el estudio bibliográfico acerca del objeto de estudio de la investigación, con el propósito de definir las características, herramientas y tecnologías de la propuesta de solución.

Métodos empíricos

Entrevista: se utilizó para identificar las necesidades existentes en la línea de Producción de Recursos Educativos del centro FORTES y de esta manera definir el objetivo que estará dirigido a la propuesta de solución.

Observación: se utilizó para identificar algunas características de la propuesta de solución, con el propósito de realizar un control adecuado de la implementación de los componentes personalizables. La guía de observación se puede consultar en el Anexo 2: Guía de observación.

El siguiente trabajo está estructurado de tres capítulos los cuales se describen a continuación:

Capítulo 1: Fundamentación teórica.

El capítulo aborda toda la fundamentación teórica de la presente investigación. Se realiza un estudio del arte de los componentes básicos para el marco de trabajo. Además, se expone la metodología, las herramientas y tecnologías, así como los lenguajes de programación, empleados en el desarrollo de los componentes personalizables para el marco de trabajo de desarrollo de multimedia.

Capítulo 2: Solución y propuesta.

Este capítulo contiene una descripción de la propuesta de solución. Agrupa el diagrama de modelo del dominio y los de clases del diseño generados a partir del problema planteado. Además, de especificar los requisitos funcionales y no funcionales que deberá cumplir la solución de la investigación.

Capítulo 3: Implementación y pruebas.

Este capítulo contiene las clases a utilizar en la implementación de la solución. Describe las pruebas de software realizadas para lograr la calidad del producto, permitiendo la satisfacción del cliente.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se aborda todo el marco teórico conceptual asociado al objeto de la investigación. Se analizan las soluciones similares para obtener las funcionalidades que servirán como base para el desarrollo de los componentes personalizables. Se describe la metodología de desarrollo de software, las herramientas y tecnologías que serán empleadas en la propuesta de solución.

1.2 Conceptos asociados al dominio del problema

A continuación se definen los términos más significativos relacionados con el objeto de estudio, para un mejor entendimiento de la propuesta de solución.

1.2.1 Multimedia

El término multimedia hace referencia al uso combinado de diferentes medios de información como texto, imagen, sonido, animación y video. Alguna de las características que presentan las multimedia son (Belloch, 2012):

- Interactividad: es la comunicación recíproca, acción y reacción existente entre la aplicación multimedia y el usuario.
- Ramificación: es la capacidad del sistema multimedia para responder a las preguntas del usuario.
- Transparencia: la tecnología tiene que permitir la utilización de los sistemas multimedia de manera sencilla y rápida.
- Navegación: es la posibilidad que tiene el usuario de desplazarse por la información de forma adecuada sin perderse por la aplicación multimedia.

1.2.2 Medios de información

Los medios de información juegan un papel fundamental dentro de cualquier software, pueden potenciar la memoria visual y auditiva, además facilita el nivel de aprendizaje del usuario. A continuación se describen los medios de información (Belloch, 2012):

Texto: el texto tiene como función principal favorecer la reflexión y profundización en los temas. Alguna de las ventajas que proporciona su inclusión en las aplicaciones multimedia es desarrollar la comprensión lectora y la fluidez verbal del usuario.

Imágenes estáticas: ilustran y facilitan la información que se desea transmitir en una multimedia.

Imágenes dinámicas, videos o animaciones: transmiten de forma visual secuencias completas de contenido, ilustrando una parte de la información que tiene sentido propio. La

animación permite un mayor control de las situaciones, a través de esquemas y figuraciones que la imagen real no refleja en los videos.

Sonido: son incorporados en las multimedia para facilitar la comprensión de la información. Estos pueden ser locuciones orientadas a completar el significado de las imágenes, música y efectos sonoros para conseguir un efecto motivador en el usuario.

Iconográficos: permiten la representación de palabras, conceptos e ideas, mediante imágenes y gráficos.

Gráfico: cualquier elemento que pueda ser presentado en la pantalla de una computadora puede emplearse como un gráfico en un documento multimedia.

Los medios de información pueden ser gestionados por componentes, razón por la cual el desarrollo de multimedia puede ser analizado como un proceso basado en componentes. Un componente es una parte reemplazable, casi independiente y no trivial de un sistema que cumple una función clara en el contexto de una arquitectura bien definida (Pressman, 2005). Entre las principales ventajas que ofrece la utilización de componentes se encuentran la reutilización en diversas aplicaciones, reducción de costos y el mejoramiento de la calidad.

1.2.3 Componente personalizable

Personalización

La personalización de aplicaciones es entendida como la capacidad de alteración dinámica con el fin de proporcionar al usuario la impresión de estar trabajando con una aplicación específicamente diseñada para dar satisfacción a sus necesidades particulares. (Cachero, 2010)

Componente personalizable

Definido los términos componente y personalización se procede a definir componente personalizable. Debido a que no existe una definición de componente personalizable los autores de la presente investigación lo definen de la siguiente forma:

Un componente personalizable es una parte reemplazable del software, le brinda al usuario la posibilidad de ajustar la interfaz acorde a sus necesidades y preferencias. Además, adapta su estructura y contenido al dispositivo de acceso.

1.2.4 Marco de trabajo

El término marco de trabajo o *framework*, hace referencia a una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. Es decir, un marco de trabajo se puede considerar como una aplicación genérica

incompleta y configurable a la que se puede añadir las últimas piezas para construir una aplicación concreta. (Aponte, 2012)

Los principales objetivos de un marco de trabajo son:

- Disminuir el tiempo del proceso de desarrollo.
- Reutilizar código ya existente.
- Promover buenas prácticas de desarrollo como el uso de patrones.

1.3 Estudio de soluciones similares

A continuación, se realiza un estudio de las soluciones similares tanto nacionales como internacionales. Con el propósito de identificar las funcionalidades bases para el desarrollo de los componentes personalizables.

1.3.1 Internacionales

A continuación se exponen las soluciones similares internacionales estudiadas:

***svBuilder*:** es una aplicación de escritorio que permite crear galerías de simples vistas, cambiar automáticamente el tamaño de las imágenes, genera miniaturas y crea código para incrustar XML y HTML. (svBuilder-Pro, 2016)

Esta herramienta está compuesta por los siguientes paneles principales:

- Fuente de imagen: se puede seleccionar cualquiera de las imágenes locales o imágenes de Flickr.
- Añadir imagen: para añadir imágenes a la galería se debe arrastrar y soltar la imagen desde el explorador de archivos, o hacer clic en “Examinar” para seleccionar las imágenes del sistema de archivos.
- Tamaño de la imagen: está compuesta por varias opciones como “Tamaño de las imágenes” permite cambiar el tamaño de las imágenes importadas a las dimensiones especificadas de forma automática. “Recortada a” redimensiona la imagen y “Cambiar” cambia la imagen en miniatura, las dimensiones y calidad.
- Lista de imágenes: Muestra todas las imágenes de la galería.

svBuilder posee las siguientes funcionalidades:

- Crear galería.
- Abrir galería existente.
- Guardar galería.
- Visualizar galería.
- Añadir imagen.
- Eliminar imágenes de la galería.
- Ordenar imágenes.
- Cargar imágenes.

- Guardar ajustes preestablecidos.

En la siguiente figura se muestra la interfaz de la aplicación.

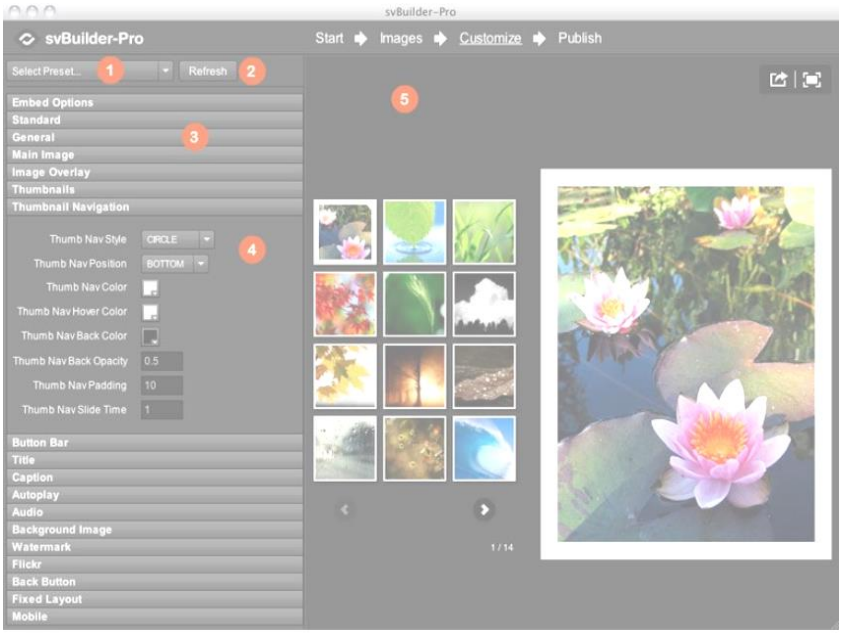


Figura 1: Pantalla de la aplicación *svBuilder*

svBuilder a pesar de ser una aplicación con grandes funcionalidades es privativa, por esta razón es de difícil acceso. Además, las galerías generadas no son adaptables a los dispositivos móviles. Así como también se requiere del complemento Adobe AIR 2.0 o superior para su instalación y en Linux el 2.6.

***Layoutit*:** es una plataforma para crear páginas web con diseño *responsive*, es decir, páginas que se adapten a cualquier pantalla alterando su estructura en función del tamaño del dispositivo que la esté visualizando. La aplicación tiene todos los elementos y componentes de Bootstrap para hacer de su interfaz más fácil sin necesidad de ser un experto en JavaScript, HTML5 o CSS3. (Layoutit, 2015)

Algunas de las características que posee *Layoutit* son las siguientes:

- Permite arrastrar y soltar los componentes Bootstrap de inicio a su propio diseño.
- Es fácil de integrar con cualquier lenguaje de programación.

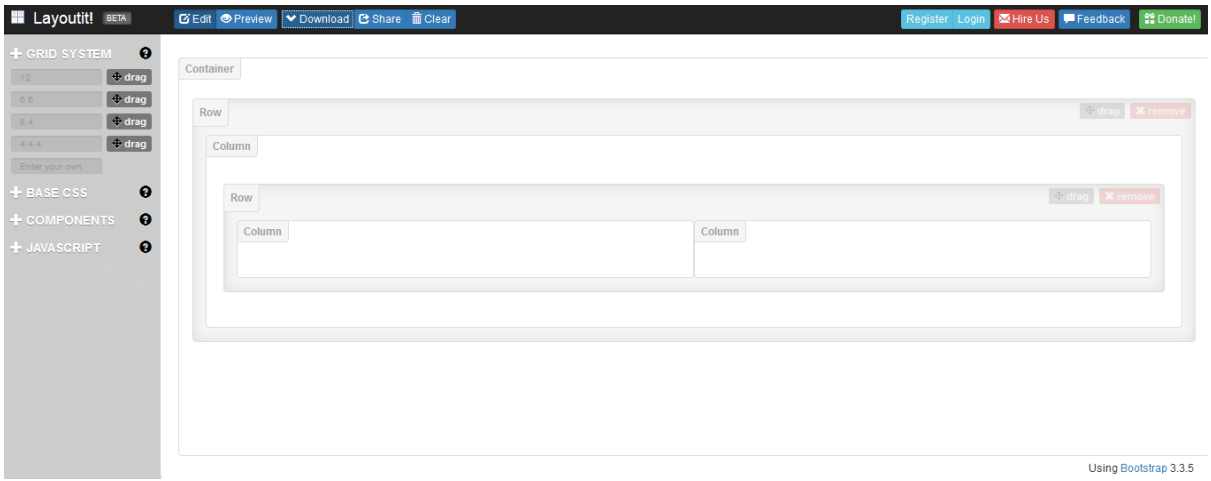


Figura 2: Pantalla de la aplicación *Layoutit*

Layoutit a pesar de permitirle al usuario crear plantillas para sus páginas web, no les permite modificar el color de las regiones y el de fondo de la plantilla. Se necesita conexión a internet para trabajar con dicha herramienta. Además, es una aplicación que al no tener incluida la tecnología AngularJS no permite crear multimedia empleando dicha tecnología.

1.3.2 Nacionales

Componentes básicos

En el proyecto Marco de trabajo de desarrollo de multimedia con tecnologías libres se desarrollaron una serie de componentes básicos, con el fin de facilitarle al usuario la confección de una multimedia y disminuir el tiempo de desarrollo de la misma.

Un **componente básico** posee un conjunto de plantillas predeterminadas que se ajustan a las necesidades básicas de los usuarios.

A continuación, se mencionan los componentes básicos existentes:

Galería de imágenes: contiene un paquete de plantillas visuales, predeterminadas para hacer más sencilla la estructuración y el diseño de la galería de imagen. Este componente posee las siguientes funcionalidades:

- Listar tema de galería de imágenes.
- Seleccionar tema de galería de imágenes.
- Adicionar descripción a la imagen.
- Ampliar imagen.

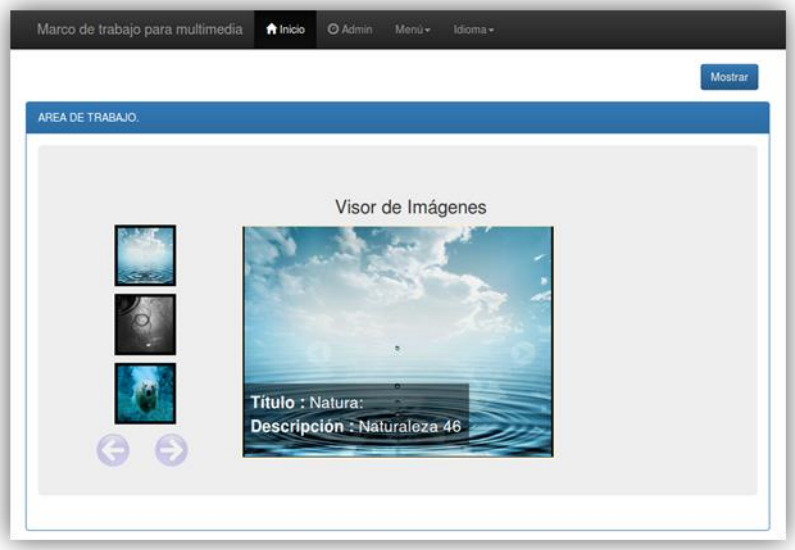


Figura 3: Componente básico Galería de imágenes

Galería de videos: contiene un paquete de plantillas visuales, predeterminadas para hacer más sencilla la estructuración y el diseño de la galería de video. Este componente posee las siguientes funcionalidades:

- Listar temas de galería de videos.
- Seleccionar tema de galería de videos.
- Adicionar descripción al video.

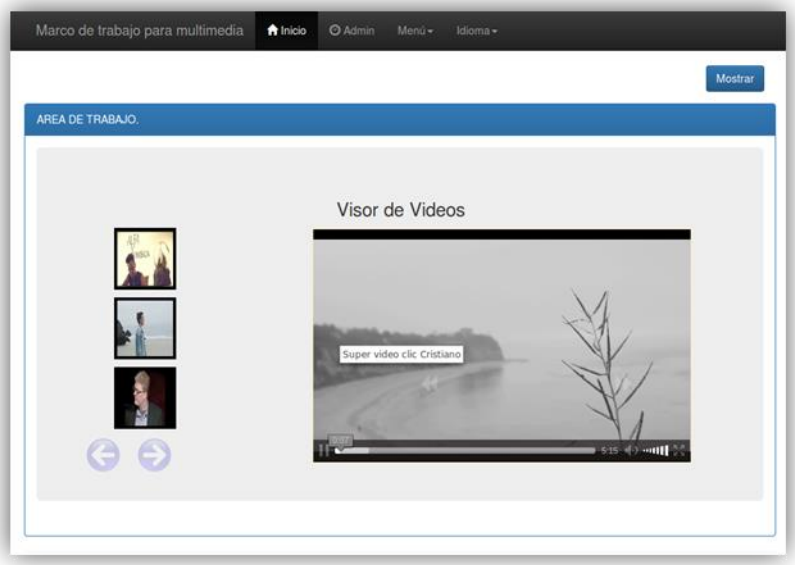


Figura 4: Componente básico Galería de videos

Glosario de términos: agrupa las palabras desconocidas por el usuario con su significado. Este componente posee las siguientes funcionalidades:

- Mostrar el glosario de términos.

- Seleccionar letra.

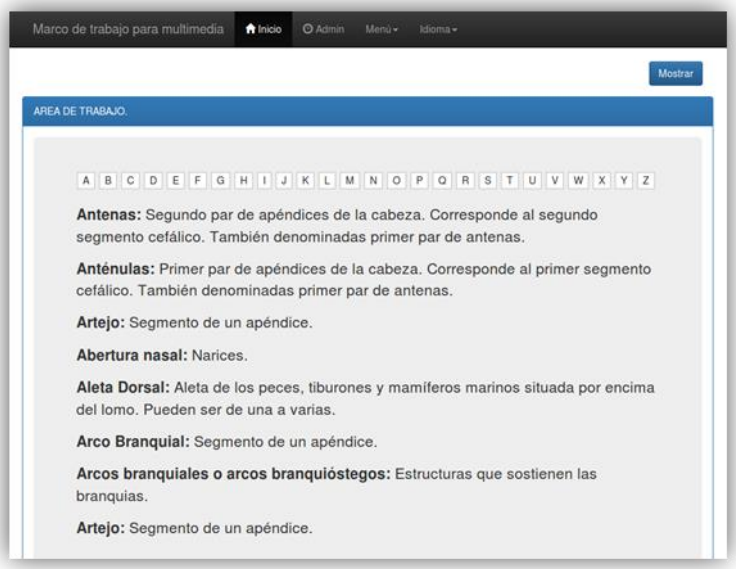


Figura 5: Componente básico Glosario de términos

Menú: contiene un paquete de plantillas visuales, predeterminadas para hacer más sencilla la estructuración y el diseño de navegación. Este componente posee las siguientes funcionalidades:

- Listar temas de menús.
- Seleccionar tema de menú.
- Realizar búsqueda de información.

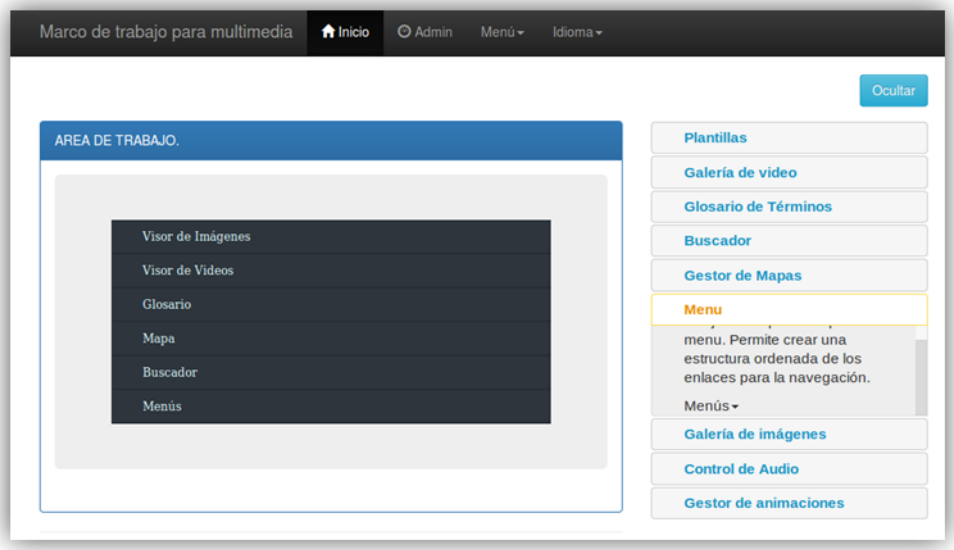


Figura 6: Componente básico Menú

Animaciones: contiene un paquete de animaciones con diferentes efectos para animar los objetos de una multimedia. Este componente posee las siguientes funcionalidades:

- Cargar imagen dentro del proyecto.
- Seleccionar tipo de animación.
- Seleccionar efecto de animación.
- Definir coordenadas de inicio y fin.
- Definir velocidad de transición.

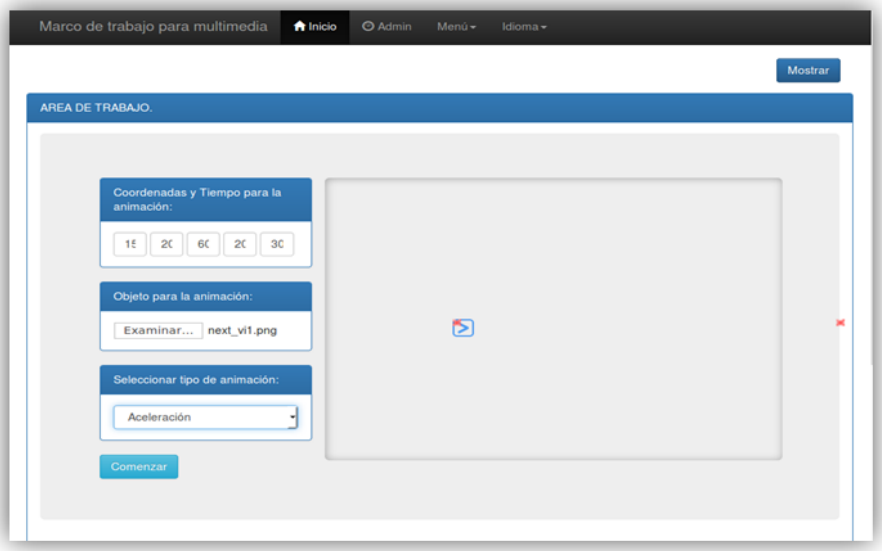


Figura 7: Componente básico Animación

Tema: permite que el usuario seleccione un tema de los existentes para crear una multimedia. A continuación, se muestran las funcionalidades que posee:

- Listar los temas.
- Seleccionar tema.

Figura 8: Componente básico Tema

Mapa: permite al usuario realizar las configuraciones necesarias para que el mapa se ajuste a sus requerimientos. A continuación, se muestran las funcionalidades que posee:

- Mover punto del mapa.
- Alejar y acercar mapa.
- Fijar punto del mapa.

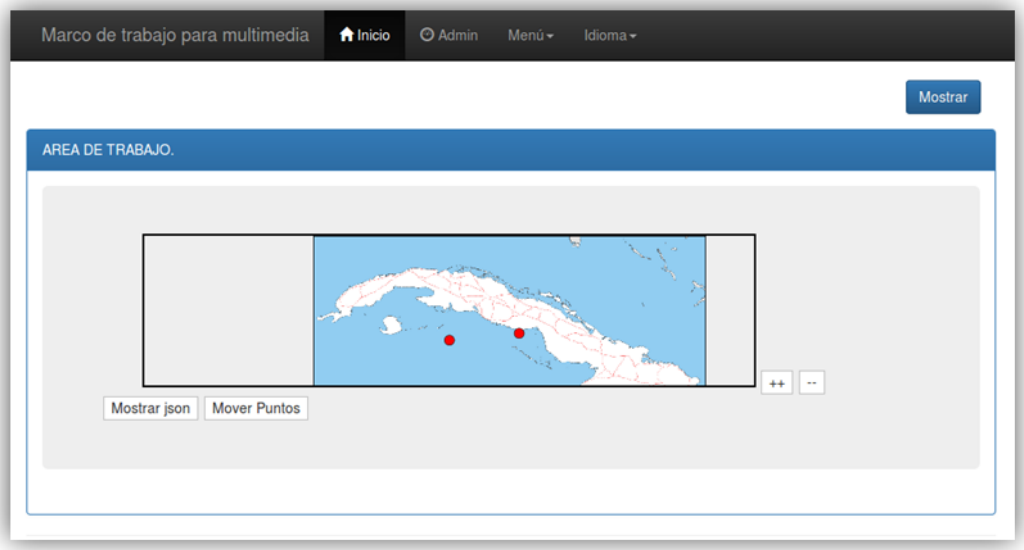


Figura 9: Componente básico Mapa

Realizado el estudio de los componentes básicos se detectaron deficiencias como la baja adaptabilidad en pantalla que poseen para los diferentes dispositivos móviles. Además, solo poseen plantillas de diseño predeterminadas, lo que trae consigo que en ocasiones no se ajusten a los requerimientos de los usuarios.

El análisis realizado permitió determinar las funcionalidades que serán utilizadas como base para el desarrollo de los componentes personalizables. A continuación, se exponen estas funcionalidades:

- Adicionar descripción de la imagen.
- Ampliar imagen.
- Adicionar descripción al video.
- Seleccionar letra.
- Mostrar glosario.
- Seleccionar efectos de animación.
- Cargar imagen.
- Mover punto en el mapa.
- Alejar y acercar mapa.

Los componentes personalizables son:

- Galería de imágenes

- Galería de videos
- Glosario de términos
- Animaciones
- Menú
- Tema
- Mapa

1.4 Metodología, tecnología y herramienta

Para el desarrollo de la propuesta de solución no fue necesario realizar un estudio de metodologías, tecnologías y herramientas debido a que la presente investigación se rige por las especificaciones expuestas en los documentos de Arquitectura de Software versión 1.1 y el Plan de desarrollo de Software versión 1.5.

1.4.1 Metodología de desarrollo de software

Para el diseño y desarrollo de proyectos de software se aplican metodologías. Las cuales son un conjunto de procedimientos, técnicas y ayuda a la documentación para el desarrollo de productos de software. A partir de la metodología seleccionada se va indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado. Se indican las personas que deben participar en el desarrollo de las actividades y el papel que deben tener. Además, se detalla la información que se debe producir como resultado de una actividad y la información para comenzarla. En el proyecto Marco de trabajo se emplea la metodología de desarrollo Proceso Unificado Ágil (AUP).

Proceso Unificado Ágil (AUP)

Constituye una versión simplificada del Proceso Unificado Racional (*Rational Unified Process*, RUP), desarrollada por Scott Ambler. Esta metodología combina procesos propios del concepto unificado tradicional con técnicas ágiles, con el objetivo de mejorar la productividad. Permitiendo así describir de manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio. AUP aplica técnicas ágiles, entre las que se incluyen (Sánchez, 2014):

- El desarrollo dirigido por pruebas.
- El modelado ágil.
- La gestión de cambios ágil.
- La refactorización de bases de datos para mejorar la productividad.

Al igual que RUP, en AUP se definen cuatro fases que se desarrollan de manera consecutiva:

1. Inicio.
2. Elaboración.

3. Construcción.
4. Transición.

AUP define 7 disciplinas, de ellas cuatro son dirigidos para la ingeniería y 3 para la gestión de proyectos. La Universidad realiza una variación definiendo nuevas disciplinas, establecidas de la siguiente forma:

1. Modelado de negocio.
2. Requisitos.
3. Análisis y diseño.
4. Implementación.
5. Pruebas internas.
6. Pruebas de liberación.
7. Pruebas de aceptación.
8. Despliegue.

Las disciplinas propuestas por AUP: gestión de configuración, gestión de proyectos y entorno, se cubrirán de acuerdo a las áreas de procesos que define CMMI-DEV en su versión 1.3. Estas quedan de la siguiente forma: gestión de la configuración, la planeación del proyecto, el monitoreo y control de proyecto.

AUP propone 9 roles en el desarrollo de un proyecto. Pero debido a la adaptación realizada por la Universidad, decide para el ciclo de vida de los proyectos tener 11 roles, de los cuales se mantienen algunos de los propuestos por AUP y unificando o agregando otros para una mejor organización del equipo de desarrollo. A continuación, se muestran los roles resultantes:

1. Jefe de proyecto.
2. Planificador.
3. Analista.
4. Arquitecto de información.
5. Desarrollador.
6. Administrador de la configuración.
7. *Stakeholder* (Cliente/Proveedor de requisitos).
8. Administrador de calidad.
9. Probador.
10. Arquitecto de software.
11. Administrador de base de datos.

1.4.2 Tecnologías y herramientas

A continuación se mencionan las tecnologías y herramientas utilizadas para la implementación de la propuesta de solución:

1.4.2.1 Lenguajes de desarrollo

Los lenguajes de programación son un conjunto de reglas, normas, símbolos y caracteres que le permite a un programador la creación de programas y resolver los mismos de manera eficaz (Hernández, y otros, 2014). Para el desarrollo de la presente investigación se utilizaron los lenguajes del lado del cliente y del servidor.

Lenguajes del lado del cliente: son aquellos que el programa reside junto a la página web en el servidor, pero es transferido al cliente para que este los ejecute. Es decir, el servidor no interviene en el proceso de crear la página web solicitada por el usuario.

Lenguajes del lado del servidor: los lenguajes son ejecutados por el servidor y lo que se envía al cliente es la respuesta o el resultado de dicha ejecución.

A continuación, se describen los lenguajes que se utilizaron:

HTML (*HyperText Markup Language*): es el elemento de construcción más básico de una página web, se usa para crear y representar visualmente una página web. Determina el contenido de la página web, pero no su funcionalidad. HTML le añade "marcado" a un texto estándar en español. "Hipertexto" se refiere a enlaces que conectan una página web con otra. Con la ayuda de HTML se pueden hacer sitios estáticos y dinámicos. (Florentino, 2016)

La versión a utilizar es **HTML5**, esta incluye una serie de elementos semánticos para dar significado a las diferentes partes de la página web como son (Lennon, 2011):

- Header: se utiliza para definir la cabecera del documento/sección.
- Footer: se emplea para delimitar el pie del documento o el final de las diferentes secciones que poco tienen que ver con el contenido de una página.
- Nav: este elemento se emplea para definir la zona de navegación con vínculos en la página web.

Para indicar que una página web sigue las reglas de HTML5 se tiene que indicar al principio del código la siguiente línea <! DOCTYPE html>.

HTML5 posee varias novedades con respecto a la anterior como son:

- La inserción de elementos multimedia con las etiquetas audio y video en los sitios web, ya que permite reproducir y controlar los audios y videos en las páginas web.
- La etiqueta canvas te permite crear un lienzo donde podrás dibujar gráficos por vectores.
- Se puede reducir la dependencia de complementos como es el de Adobe Flash.
- El elemento input ahora permite tipos de datos como son los *number*, *url*, entre otros.
- Permite que las aplicaciones web puedan trabajar sin conexión.

CSS (*Cascading Style Sheets*): es un lenguaje usado para definir la presentación semántica de un documento estructurado escrito en HTML o XHTML. Se crea con el principal objetivo de separar el contenido de la forma en que se visualiza la página web, permitiéndoles a los diseñadores un mayor control sobre las apariencias de sus páginas. (Pérez, 2009)

La versión a utilizar es **CSS3**, la misma está divida en módulos, aunque contiene especificaciones de CSS.

Las características principales que se destacan en CSS3 es mayor control sobre el estilo de los elementos de nuestra página web y mayor número de efectos visuales. (CSS3, 2016)

Ventajas de usar CSS3:

- Código más simple para muchas tareas.
- Mayores opciones de gráfica.
- CSS3 es más utilizado por los usuarios.

Alguno de los módulos más importantes de CSS3 es:

- Selectores
- Box Model
- Efectos de textos
- Animaciones
- Interfaz del usuario
- Disposición de columnas múltiples

JavaScript: es un lenguaje de programación interpretado por el navegador web, es compatible con la mayoría de los navegadores modernos (Chrome, Firefox, Internet Explorer, Safari y Opera). Se utiliza para crear páginas web dinámicas, desarrollado por Netscape. Es un lenguaje de script multiplataforma, sencillo, de rápida ejecución, consume poca memoria, fácil de integrar a las páginas web, tiene gran cantidad de efectos visuales. JavaScript permite añadir gran cantidad de efectos visuales y utiliza el DOM (*Document Object Model)* para modificar el HTML con mayor facilidad sin tener que recargar todo el documento. (Scholz, 2015)

Para la implantación de la propuesta de solución se utilizó el lenguaje JavaScript del lado del cliente y del servidor.

1.4.2.2 Generador de proyecto

Un generador es un complemento que se puede ejecutar mediante un comando.

Yeoman: permite generar proyectos web con un solo comando, estos pueden ser de todo tipo Ruby, PHP, JavaScript, etc. Con el propósito de crear proyectos con el *framework* AngularJS se creó un “generador”. Un generador es un complemento que se puede ejecutar con el comando “yo”. A través del mismo se promueve el “flujo de trabajo Yeoman”. Este

flujo de trabajo comprende tres tipos de herramientas para mejorar la productividad y satisfacción en la construcción de una aplicación web. (Yeoman, 2016)

Estas herramientas son:

- La herramienta de andamios (yo[1]).
- La herramienta de construcción (Grunt, Gulp).
- El gestor de paquetes (como Bower).

La versión de Yeoman a utilizar en el desarrollo de los componentes personalizables es la 1.6.0.

Bower: es un módulo de Nodejs que puede administrar los componentes que contienen HTML, CSS, JavaScript, fuentes e incluso archivos de imagen. Es un sirviente, con solo indicarle la biblioteca o *framework* que se necesita, él se encarga de descargar las versiones correctas de los paquetes y sus dependencias. Tiene además opciones de búsqueda, actualización y eliminación de assets (Bower, 2012). La versión de Bower a utilizar es la 1.7.7.

Un ejemplo de lo descrito anteriormente es: si se comienza un nuevo proyecto y se va a utilizar jQuery, AngularJS, Bootstrap, Angular-Bootstrap y MomentJS, solo hay que escribir la siguiente instrucción: “bower install jquery angular bootstrap angular-bootstrap momentjs”.

Este comando realiza 3 opciones:

- Buscar en su base de datos online la existencia de estos assets.
- Descargarlos hacia la cache (.cache/bower en Linux, %appdata%/Local/bower en Windows) en caso que no se haya descargado.
- Copiarlos hacia el proyecto.

Grunt: es un módulo de Nodejs que permite configurar tareas automáticas y así ahorrar tiempo en el desarrollo y despliegue de aplicaciones web. Alguna de las tareas más comunes a automatizar son (Grunt, 2016):

- Concatenación de archivos (CSS, JS).
- Minificación de archivos (CSS, JS).
- Optimización de imágenes.
- Compilación (SASS -> CSS y otros).
- Pruebas unitarias.

Una de las ventajas que posee Grunt es que es JavaScript al igual que los complementos y utiliza Nodejs para su ejecución, lo que le permite ser multiplataforma. La versión de Grunt a utilizar en el desarrollo de los componentes es 0.1.13.

[1] yo: es un comando de Yeoman.

NPM: es un gestor de paquetes para JavaScript que le permite a los desarrolladores compartir y reutilizar paquetes de código. Además, permite que la actualización del código a compartir sea fácil (NPM, 2016). La versión a emplear en el desarrollo de los componentes personalizables es la 2.5.1.

1.4.2.3 Marcos de trabajo

AngularJS: es un *framework* JavaScript de código abierto desarrollado y mantenido por Google. Implementa el patrón de diseño Modelo Vista Controlador (MVC), para crear aplicaciones de una sola página (single-page-applications). AngularJS extiende el tradicional HTML con etiquetas (directivas) propias. Posee un sistema de *databinding* extensible haciendo uso de las directivas y los filtros, lo que propicia que la capa de presentación el código sea más simple y organizado. AngularJS pretende que los programadores que lo usen generen un HTML entendible para aquellas personas que no tengan mucho conocimiento informático y que una vez lo lean sepan que es lo que hace y para qué sirve la aplicación creada (Alvarez, 2014). La versión de AngularJS a utilizar para el desarrollo de los componentes personalizables es la 1.4.9.

Características de AngularJS:

- Fácil comprensión para los que comienzan a usarlo pues ofrece características sofisticadas para desarrolladores con necesidades complejas.
- El código de aplicaciones creadas con AngularJS siempre está organizado por Modelos, Vistas, Controladores y opcionalmente Servicios.

Ventajas de AngularJS:

- Potente sistema de *templating* incluido en el mismo *framework*.
- Sincronización entre vistas y modelos para crear páginas one-page.
- Uso de directivas para la creación de nuevos atributos o nuevas etiquetas HTML.
- Uso de filtros para alterar la presentación de datos.
- Uso de Servicios que se encargan de la comunicación con el servidor para la consulta de datos.

Jasmine: es un marco de desarrollo impulsado por el comportamiento de JavaScript para probar aplicaciones de AngularJS. No depende de ningún otro *framework* de JavaScript, así como tampoco requiere un DOM. Jasmine posee una sintaxis limpia por lo que se puede escribir fácilmente pruebas (Jasmine, 2016). La versión a utilizar para el desarrollo de los componentes personalizables es la 2.4.1.

Express: es un *framework* de aplicaciones web Node.js mínima y flexible, que proporciona un conjunto solido de características para las aplicaciones web y móviles. Express define una tabla de enrutamiento que se utiliza para llevar a cabo una acción diferente, basada en

el método HTTP y URL. Además, permite representar dinámicamente páginas HTML (Express, 2016). La versión a utilizar para el desarrollo de los componentes personalizables es la 4.13.4.

Bootstrap: es un *framework* creado inicialmente por *Twitter* y posteriormente liberado bajo la licencia MIT, para el desarrollo de interfaces web. Incluye HTML y CSS basado en el diseño de plantillas para la tipografía, formularios, botones, tablas, módulos, carruseles de imágenes y muchas otras. Es una herramienta compatible con todos los navegadores modernos (Chrome, Firefox, Internet Explorer, Safari, and Opera). Es adaptable al tamaño de cualquier dispositivo donde se visualice, como los dispositivos móviles, tablets y ordenadores de sobremesa. (Bootstrap, 2016)

La versión de Bootstrap a utilizar es la 3.1.1, esta consta de varias características importantes como:

- Tiene un soporte casi completo con HTML5 y CSS3, flexibilizando la creación de aplicaciones web.
- Permite utilizar un GRID de 12 columnas donde colocar el contenido, posibilitando crear diseños web *responsive* de una forma más fácil.
- Permite la inserción de imágenes responsive con solamente ponerle la clase “img-responsive” a las imágenes, estas se adaptan al tamaño deseado.

1.4.2.4 Entorno integrado de desarrollo

Un Entorno Integrado de Desarrollo (IDE por sus siglas en inglés), es un entorno de programación que ha sido empaquetado como un programa de aplicación. Es decir, un IDE se compone de un editor de código de programación, un compilador, un intérprete, un depurador y un constructor de interfaz gráfica. (Ricardo Barrera Urieta, 2014)

A continuación se menciona el IDE a utilizar para el desarrollo de la propuesta de solución:

WebStorm: es un IDE de JavaScript multiplataforma desarrollado por JetBrains[2]. Utilizado para desarrollar aplicaciones web con soporte para las tecnologías JavaScript, Nodejs, HTML y CSS. Posee finalización de código inteligente, detección de errores en el mismo momento en que se escribe. WebStorm tiene soporte para el desarrollo web con *framework* como AngularJS, React, Meteor, entre otros. La versión que se utilizará es la 11.0, algunas de las características principales son (WebStorm, 2016):

- Producir código de alta calidad de manera más eficiente, gracias a la finalización de código inteligente, detección de errores y la navegación de gran alcance sobre la marcha.

[2] JetBrains: es una empresa de desarrollo de software cuyas herramientas que están dirigidas a los desarrolladores de software y administradores de proyectos.

- Funciona en Windows, Mac OS o Linux con una única clave de licencia.
- Es un software con todas las funciones para construir sitios web que usan herramienta como el depurador, VCS, en terminales y otras herramientas.
- Ofrece soporte para JavaScript, Node.js, ECMAScript 6, mecanografiado, CoffeeScript, Menos, Sass, Stylus y Dardo.
- Combina los mejores resultados de la finalización de código para todos los métodos, funciones, módulos, variables y clases definidas.

1.4.2.5 Intérprete de JavaScript

Un motor JavaScript (también conocido como intérprete de JavaScript) es la parte del navegador que interpreta y ejecuta el código escrito en el lenguaje de programación JavaScript. (JavaScript Engines, 2015)

Nodejs: es un intérprete de JavaScript, construido sobre el motor JavaScript v8 de Chrome. Posee un entorno de ejecución multiplataforma de código abierto para el desarrollo del lado del servidor y las aplicaciones de red escalables, contiene un entorno de programación dirigido por eventos, basado en el lenguaje de programación ECMAScript. La versión que se va a utilizar es la 0.12.0 la cual presenta las siguientes ventajas (Nodejs, 2015):

- Está basado en eventos, así que toda la filosofía asíncrona utilizada con AJAX en el cliente se puede pasar al servidor.
- Permite utilizar el mismo lenguaje (JavaScript) tanto en el cliente como en el servidor.
- El desarrollo es más rápido.
- La ejecución de test de unidad se puede hacer más rápido.

1.4.2.6 Herramienta CASE

Las herramientas CASE (*Computer Aided Software Engineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo del costo, compilación automática, documentación o detección de errores. (Flores, 2013)

Visual Paradigm: es una herramienta CASE, que soporta el modelado mediante UML. Proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. (Paradigm, 2015)

La versión a utilizar en la presente investigación es la 8.0. Esta versión incluye la funcionalidad de crear y especificar perfiles UML, resulta de vital importancia para la implementación de herramientas.

1.5 Conclusiones del capítulo

Se elaboró el marco teórico que sustenta la presente investigación, para un mejor entendimiento de la aplicación que se desea desarrollar. A partir del estudio de soluciones similares se pudo determinar las funcionalidades genéricas de los componentes básicos, que serán utilizadas como base para el desarrollo de los componentes personalizables. Se describe la metodología de desarrollo de software AUP definida por el proyecto para generar los artefactos ingenieriles de la propuesta de solución. En la implementación de la propuesta de solución se decide utilizar las tecnologías AngularJS y Bootstrap, seleccionándose las herramientas WebStorm, Nodejs y Visual Paradigm para el modelado UML.

Capítulo 2: Solución y propuesta

2.1 Introducción

En el presente capítulo se muestra el modelo de dominio para relacionar los conceptos asociados a la investigación. Se realiza el levantamiento y especificación de los requisitos funcionales y no funcionales, para el desarrollo de los componentes personalizables. Se plantea además la propuesta de solución al problema principal de la presente investigación. Se describe la arquitectura a utilizar y los patrones de diseños empleados para el desarrollo de los componentes.

2.2 Modelo de dominio

Un modelo de dominio es una representación visual de las clases conceptuales significativas en un dominio (Larman, 2003). Es el artefacto más importante que se crea durante el análisis orientado a objetos, debido a que brinda un mejor entendimiento de los conceptos manejados en una aplicación informática.

A continuación, se mencionan los conceptos que serán empleados en el modelo de dominio para el desarrollo de los componentes personalizables.

Medios de información: son los elementos que conforman una multimedia. Estos medios son:

- Texto: está presente en toda la multimedia, describen las imágenes y videos de las respectivas galerías. Es utilizado en el menú, en el glosario de términos para definir las palabras y en el mapa para describir el lugar señalado por el usuario.
- Imagen: permite transmitir una información y logra una mayor interactividad con el usuario. Estas imágenes pueden ser visualizadas en formatos jpg, png o gif.
- Sonido: permite expresar una información y hacer más interactiva la multimedia.
- Video: es una secuencia de imágenes animadas transmitidas en un tiempo determinado. Estos videos se pueden visualizar a través de los formatos mpg, avi, mp4 o webm.
- Animaciones: permite animar los elementos que están presentes en una multimedia.
- Iconográfico: es utilizado en el componente mapa para diferenciar los puntos que serán ubicados en el mismo.

Multimedia: es un sistema informático que combina diferentes medios de comunicación texto, imágenes, audio, video y animación, con el propósito de transmitir un conocimiento de manera dinámica al usuario.

Componente personalizable: son los componentes que cuenta con los requerimientos y funcionalidades de los usuarios para desarrollar una multimedia. Estos componentes son:

- Galería de imágenes: permite al usuario visualizar las imágenes de acuerdo a sus requerimientos.
- Galería de videos: permite a los usuarios visualizar los videos de acuerdo a sus requerimientos.
- Glosario de términos: ayuda a entender los términos deseados por el usuario, empleados en una multimedia.
- Animaciones: permite establecer efectos de animaciones a una imagen.
- Tema: componente que permite definir el espacio de trabajo para el desarrollo de una multimedia.
- Menú: permite la navegabilidad dentro de una multimedia y brinda la posibilidad de establecer un estilo para el componente.
- Mapa: permite conocer la localización y descripción de un lugar ubicado por el usuario en el mapa.

Marco de trabajo: es una estructura de software compuesta por componentes personalizables que permiten el desarrollo de una aplicación.

Entorno de desarrollo: es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software. Normalmente consiste de un editor de código fuente y herramientas de construcción automáticas. Muchos garantizan realizar la construcción o ensamblaje del software de manera visual permitiendo integrar componentes y disminuyendo la escritura de código fuente

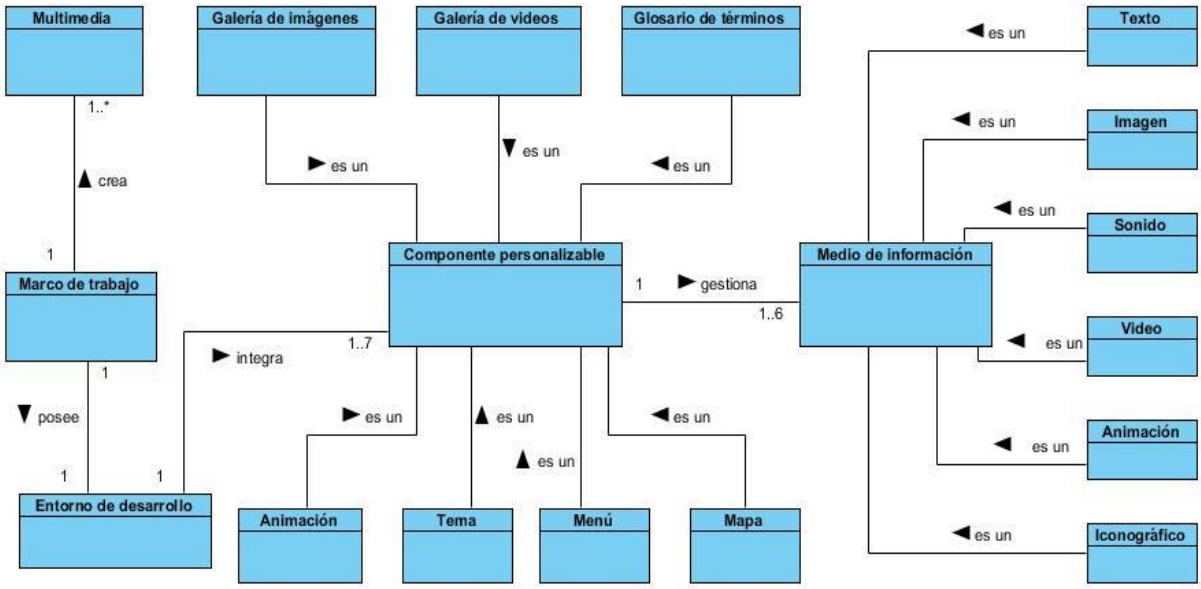


Figura 10: Diagrama de dominio de la propuesta de solución

2.3 Descripción de la propuesta de solución

La propuesta de solución consiste en desarrollar los siguientes componentes personalizables: galería de imágenes, galería de videos, glosario de términos, animaciones, tema, menú, y mapa.

Estos componentes les permitirán a los usuarios contar con plantillas de diseño cuyos elementos sean configurables. Por lo que obtendrán un producto que se ajuste a sus requerimientos, podrán ser ejecutados en cualquier sistema operativo y en diferentes dispositivos. También pueden ser reutilizados en productos que dentro de su arquitectura tecnológica empleen AngularJS.

2.4 Requisitos de software

Se entiende por requisito de software la condición o capacidad que un sistema, producto, servicio o componente debe satisfacer o poseer para cumplir un contrato, norma, especificación u otros documentos formalmente impuestos. Los requisitos incluyen las necesidades, deseos, expectativas cuantificadas y documentadas del patrocinador, el cliente y otros interesados. (PMBOK, 2008)

Existen dos tipos de requisitos: funcionales y no funcionales. A continuación, se exponen cada uno de estos requisitos.

2.4.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y como el sistema debe comportarse en distintas situaciones. En algunos casos estos requisitos también pueden declarar explícitamente lo que el sistema no debe hacer. (Sommerville, 2007)

La propuesta de solución de la presente investigación debe de satisfacer los requisitos funcionales siguientes:

RF1. Incluir descripción de la imagen.
RF2. Incluir título de la imagen.
RF3. Configurar posición de los thumbnails.
RF4. Incluir cantidad de columnas de los thumbnails.
RF5. Incluir cantidad de filas de los thumbnails.
RF6. Definir dimensiones de la imagen.
RF7. Incluir título de la galería.
RF8. Seleccionar color de los frames de los thumbnails.
RF9. Seleccionar color de los frames de la imagen principal.
RF10. Modificar fondo de la galería.

RF11. Modificar los bordes de la galería.
RF12. Eliminar imágenes.
RF13. Actualizar galería.
RF14. Incluir imagen.
RF15. Modificar formato de letra.
RF16. Incluir los términos con sus descripciones.
RF17. Definir orientación de las letras.
RF18. Guardar glosario.
RF19. Modificar término y descripción.
RF20. Incluir letra.
RF21. Eliminar letra.
RF22. Definir la trayectoria de movimiento.
RF23. Configurar efectos de animación.
RF24. Incluir regiones.
RF25. Seleccionar el fondo del tema.
RF26. Seleccionar el color de las regiones.
RF27. Incluir título del video.
RF28. Definir las dimensiones del video.
RF29. Eliminar videos.
RF30. Seleccionar color de los frames del video.
RF31. Incluir descripción del video.
RF32. Incluir título del enlace.
RF33. Incluir vínculo del enlace.
RF34. Definir color del menú.
RF35. Definir estilo del menú.
RF36. Incluir mapa.
RF37. Incluir punto al mapa.
RF38. Eliminar punto del mapa.
RF39. Incluir descripciones del lugar señalado por el punto.
RF40. Incluir vínculo al punto.

2.4.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. En si se refiere a la fiabilidad, al tiempo de respuesta y la capacidad de almacenamiento que proporciona el sistema. A menudo son aplicados al sistema en su totalidad, aunque normalmente se aplican a características o servicios individuales del sistema. (Sommerville, 2007)

- **Usabilidad**

RNF1. Navegadores para acceder: Mozilla Firefox 12.0 o superior y Google Chrome 15.0 o superior.

- **Soporte**

RNF2. Los componentes deben responder correctamente ante cambios o mejoras en sus funcionalidades.

- **Documentación de usuarios en línea y ayuda del sistema**

RNF3. Los componentes deben estar acompañados de un documento que refleje la descripción de sus funcionalidades y cómo usarlos.

RNF4. Los componentes deben estar acompañados de un documento que refleje los pasos para la instalación de los componentes.

- **Requisitos de licencia**

RNF5. Las librerías y tecnologías utilizadas para la confección de cada componente deben estar bajo alguna de las siguientes licencias: *General Public License* (GPL) o *Massachusetts Institute of Technology* (MIT).

- **Portabilidad**

RNF6. Los componentes deben ser independientes de plataforma. Deben de ejecutarse tanto en Microsoft Windows como en GNU/Linux y Mac OS.

- **Reusabilidad**

RNF7. Los componentes deben ser reutilizables en otras aplicaciones, brindando para ello una interfaz de comunicación.

- **Escalabilidad**

RNF8. Cada componente debe responder correctamente ante la aplicación del diseño arquitectónico, de datos o procedimental sin afectar más de dos clases.

2.4.3 Historias de usuario

Las Historias de usuario son la propuesta de las metodologías agiles para la especificación de los requisitos del cliente. Se escribe desde el punto de vista del usuario del sistema y usando su vocabulario. (Datos, 2013)

En la siguiente tabla se muestra un ejemplo de Historias de usuario donde se explican cada uno de los campos a llenar. Para la realización de las mismas se tuvo en cuenta el plan de riesgo relacionado al proyecto. Este plan se puede consultar en el Anexo 4: Plan de riesgo.

Tabla 1. Ejemplo de Historias de usuario

<table>
<tr><td colspan="2"></td></tr>
<tr><td>Debe poner el número que se le vaya a asignar a la HU, si se cuenta con 56 HU pues se deben poner los números consecutivos para cada HU y no repetir ningún número en las HU.
Número:</td><td>Debe poner el nombre del requisito correspondiente a la HU en cuestión. Recuerde que se recomienda describir una HU por cada requisito identificado.
Nombre del requisito:</td></tr>
<tr><td>Debe poner el nombre de la persona que programará la HU en cuestión, no de la persona que la está describiendo.
Programador:</td><td>Debe poner el número de la iteración en la cual se programará la HU en cuestión. En caso de no tener identificado en el momento de describir la HU este dato se pondrá N/A.
Iteración Asignada:</td></tr>
<tr><td>Debe poner la prioridad que tiene la HU en cuestión. Esta prioridad se obtiene luego de analizar en conjunto con el Arquitecto y/o Líder de desarrollo la prioridad en cuanto a desarrollo que tiene la HU en cuestión. En caso de no tener identificado en el momento de describir la HU este dato se pondrá N/A, si ya se tiene identificado se pone en: Alta, Media o Baja.
Prioridad:</td><td>Debe poner el tiempo estimado en días en el cual se programará la HU en cuestión. En caso de no tener identificado en el momento de describir la HU este dato se pondrá N/A.
Tiempo Estimado:</td></tr>
<tr><td>Debe hacer referencia a los riesgos identificados en el Plan de Riesgos que se correspondan con la HU en cuestión. En caso de no tener identificado en el momento de describir la HU este dato se pondrá N/A.
Riesgo en Desarrollo: [Hacer referencia a los riesgos identificados en plan de riesgos]</td><td>Debe poner el tiempo real en días en el cual se programará la HU en cuestión. En caso de no tener identificado en el momento de describir la HU este dato se pondrá N/A.
Tiempo Real:</td></tr>
<tr><td colspan="2">Debe quedar descrita la HU de forma tal que el desarrollador o cualquier usuario entiendan el flujo de la misma.
Descripción:
El objetivo debe tener la siguiente estructura: infinitivo + acción + elemento.
1- Objetivo:</td></tr>
</table>

Las acciones para lograr el objetivo (precondiciones y datos) deben dejar bien reflejado las condiciones necesarias que se deben realizar antes de realizar la HU en cuestión, debe tener la siguiente estructura: para + infinitivo + los datos de + elemento + hay que. **2- Acciones para lograr el objetivo (precondiciones y datos):** El flujo de la acción a realizar debe dejar bien reflejado cada paso a seguir para ejecutar satisfactoriamente la acción en cuestión, de igual forma debe reflejar qué hacer cuando ocurre algún error. **3- Flujo de la acción a realizar:**
Debe ubicar opcionalmente las observaciones que se tengan identificadas respecto a la HU en cuestión. **Observaciones:**
El prototipo de interfaz debe corresponder al diseño lo más parecido posible que se realiza antes de desarrollar la HU en cuestión. Debe servir de guía para el posterior desarrollo de la misma. **Prototipo de interfaz:**

A continuación se muestra las Historias de usuario de solo cuatros requisitos de la Galería de imágenes, las demás se pueden consultar en el Anexo 5: Historias de usuario.

Tabla 2. Historias de usuario. Incluir descripción de la imagen

Número:1	**Nombre del requisito:** Incluir descripción de la imagen
Programador: Susana Becerra Rodriguez	**Iteración Asignada:** 1ra
Prioridad: Media	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: 3, 5, 8, 10, 11	**Tiempo Real:** 2 días
Descripción: 1. **Objetivo:** Permitir incluir una descripción para facilitar la comprensión de la imagen 2. **Acciones para lograr el objetivo (precondiciones y datos):** Para incluir la descripción de la imagen hay que: Cargar la imagen 3. **Comportamientos válidos y no válidos (flujo central y alternos):** 4. **Flujo de la acción a realizar:** El componente debe permitirle al usuario introducir un texto para describir la imagen. El componente permite guardar la descripción realizada por el usuario.	
Observaciones: Incluir descripción para describir una imagen para la Galería de imagen y un video para la Galería de video	
Prototipo elemental de interfaz gráfica de usuario:	

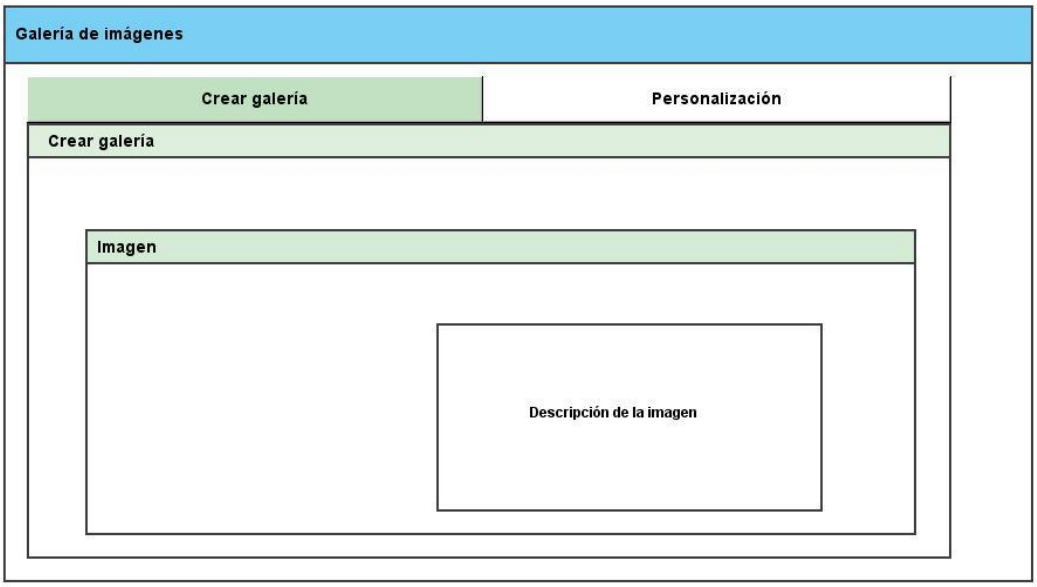


Tabla 3: Historias de usuario. Incluir título de la imagen

Número: 2	**Nombre del requisito:** Incluir título de la imagen
Programador: Susana Becerra Rodriguez	**Iteración Asignada:** 1ra
Prioridad: Baja	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: 3, 5, 8, 10, 11	**Tiempo Real:** 2 días

Descripción:

1. **Objetivo:**

Permitir incluir un título a la imagen para nombrarla

2. **Acciones para lograr el objetivo (precondiciones y datos):**

Para incluir el título de la imagen hay que:

Cargar la imagen

3. **Comportamientos válidos y no válidos (flujo central y alternos):**
4. **Flujo de la acción a realizar:**

El componente debe permitirle al usuario introducir un texto que permita nombrar la imagen.

El componente permite guardar el título de la imagen definido por el usuario.

Observaciones: : Incluir título para nombrar una imagen para la Galería de imagen y un video para la Galería de video

Prototipo elemental de interfaz gráfica de usuario:

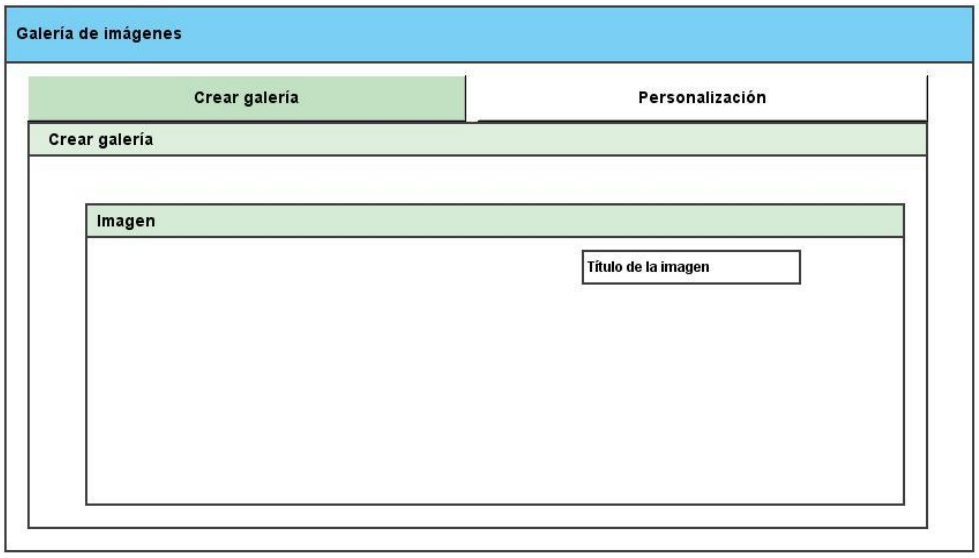


Tabla 4. Historias de usuario. Configurar posición de los thumbnails

Número: 3	**Nombre del requisito:** Configurar posición de los thumbnails.
Programador: Susana Becerra Rodriguez	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 7 días
Riesgo en Desarrollo: 3, 5, 7,8, 9, 10, 11	**Tiempo Real:** 5 días

Descripción:

1. **Objetivo:**

Permitir configurar la posición de los thumbnails para que el usuario posicione los mismos de acuerdo a su requerimiento.

2. **Acciones para lograr el objetivo (precondiciones y datos):**

Para configurar la posición de los thumbnails hay que:

Cargar la imagen

3. **Comportamientos válidos y no válidos (flujo central y alternos):**
4. **Flujo de la acción a realizar:**

El componente debe permitirle al usuario seleccionar la posición de los thumbnails.

Observaciones: La posición de los thumbnails es para que el usuario seleccione right, left, bottom, top en dependencia de cómo él quiere que estos se visualicen según sus preferencias. Esta funcionalidad solo es permitida en las galerías imágenes y video.

Prototipo elemental de interfaz gráfica de usuario:

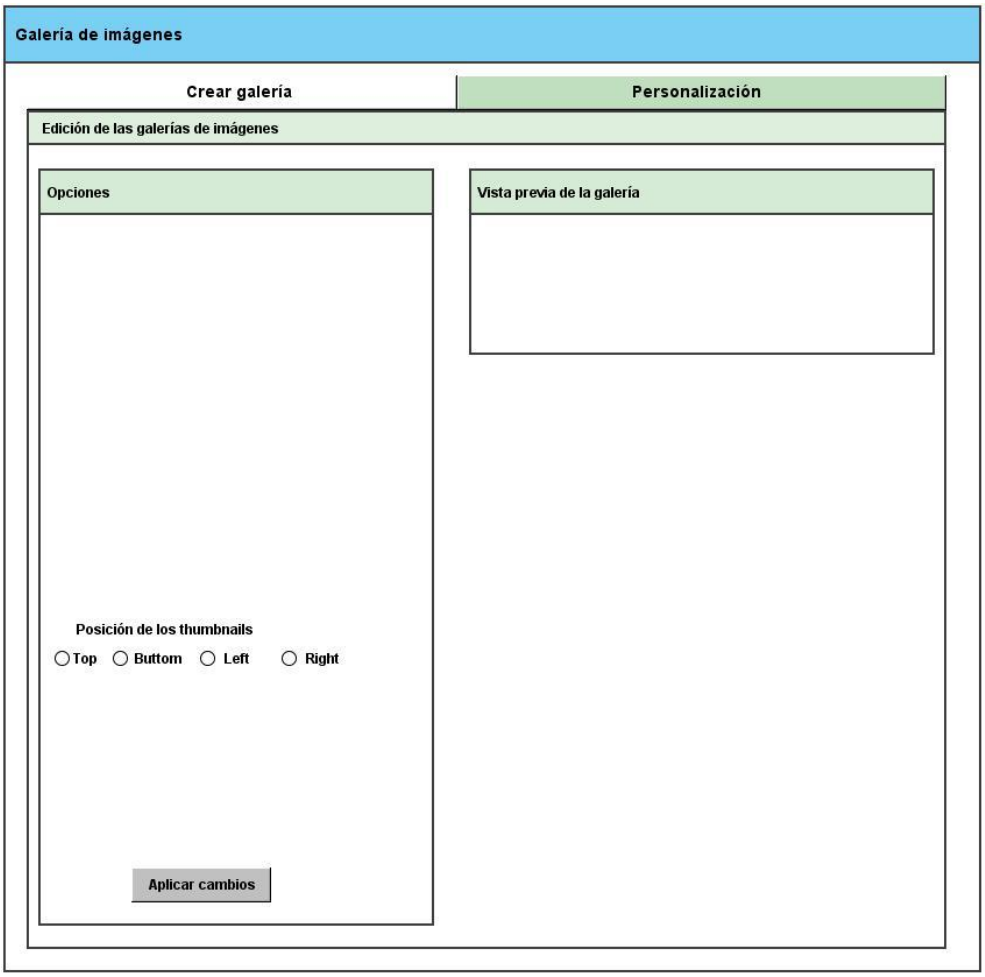


Tabla 5. Historias de usuario. Incluir cantidad de columnas de los thumbnails

Número: 4	**Nombre del requisito:** Incluir cantidad de columnas de los thumbnails
Programador: Susana Becerra Rodriguez	**Iteración Asignada:** 1ra
Prioridad: Baja	**Tiempo Estimado:** 3 días
Riesgo en Desarrollo: 3, 5, 8, 9, 10, 11	**Tiempo Real:** 2 días

Descripción:

1. **Objetivo:**

Permitir incluir cierta cantidad de columnas a los thumbnails para que el usuario pueda definir las columnas a visualizar.

2. **Acciones para lograr el objetivo (precondiciones y datos):**

Para incluir columnas a los thumbnails hay que:

Cargar las imágenes

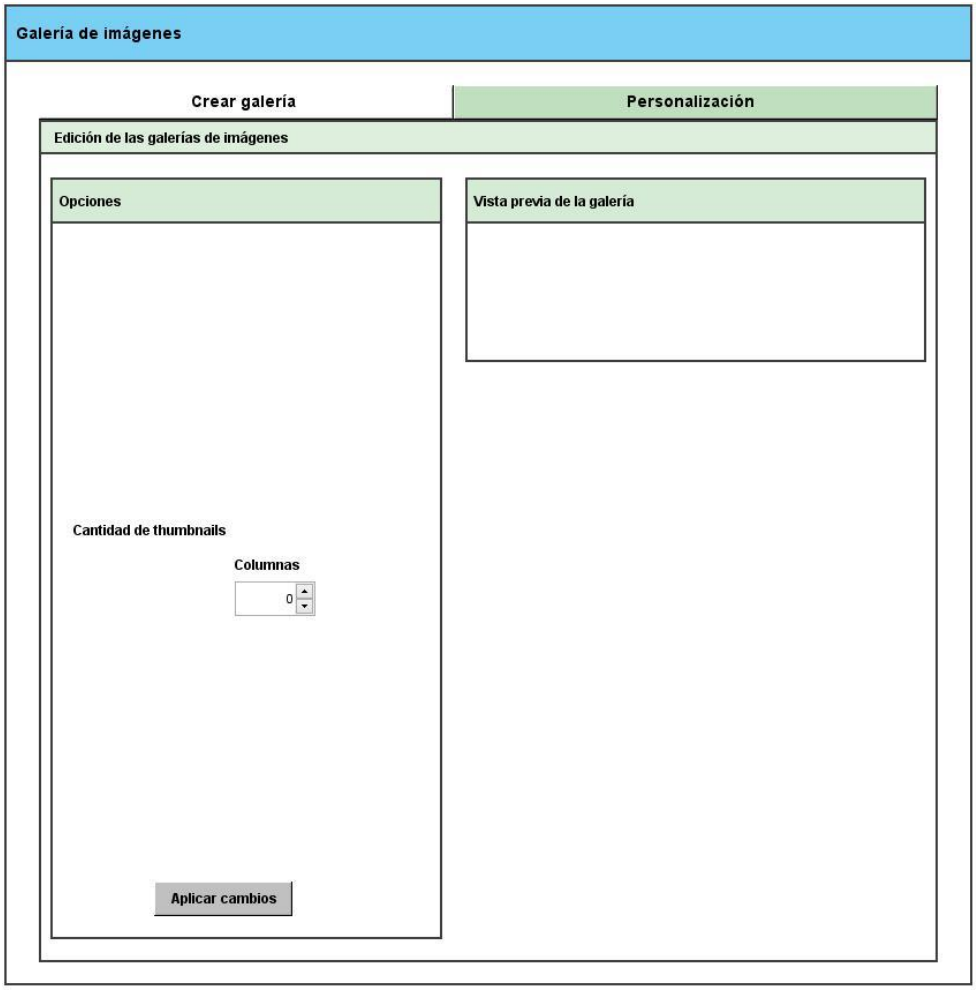
Introducir el número de columnas que se va a utilizar

3. **Comportamientos válidos y no válidos (flujo central y alternos):**
4. **Flujo de la acción a realizar:**

El componente debe permitirle al usuario introducir el número de columnas deseadas.

Observaciones: La cantidad de columnas de los thumbnails es para que el usuario pueda definir las columnas. El componente solo le permitirá al usuario definir 6 columnas si se selecciona el tipo de galería right o left. Esta funcionalidad solo es permitida en las galerías imágenes y video.

Prototipo elemental de interfaz gráfica de usuario:



2.5 Descripción de la arquitectura

La arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software. (Fuentes, 2012)

La arquitectura de la propuesta de solución está determinada por el uso de la tecnología AngularJS. Permitiendo el desarrollo de aplicaciones web siguiendo la arquitectura Modelo Vista Controlador (MVC).

El Modelo Vista Controlador es un patrón arquitectónico de software para el desarrollo de aplicaciones web. Este aísla la lógica de la aplicación de la capa de interfaz de usuario. El controlador recibe las peticiones de la aplicación y luego trabaja con el modelo para

elaborar los datos que necesita la vista. La vista utiliza los datos preparados por el controlador para generar una respuesta final. (Tutorialspoint, 2016)

El MVC se puede representar gráficamente como:

- El modelo: es responsable de la gestión de datos de la aplicación. Responde a la solicitud de la vista y con las instrucciones del controlador de actualización de sí mismo.
- La vista: es una presentación de los datos en un formato particular, desencadenada por la decisión del controlador para presentar los datos.
- El controlador: responde a la entrada del usuario y realiza interacciones sobre los objetos del modelo de datos. Este recibe la entrada, valida y a continuación realiza las operaciones de negocios que modifican el estado del modelo de datos.

Algunas de las ventajas del MVC son (Portillo, 2014):

- La separación clara entre los componentes de un programa, permitiendo su implantación por separado.
- La Interfaz de Programación de Aplicaciones (API) muy bien definida. Lo cual permite que cualquiera que utilice el API, podrá reemplazar el Modelo, la Vista o el Controlador sin ninguna dificultad.
- Conexión entre el Modelo y sus Vistas dinámicas; se produce en tiempo de ejecución y no en tiempo de compilación.

2.6 Patrones de diseño

Un patrón de diseño es una pareja de problema o solución con un nombre y es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Los patrones no se proponen descubrir ni expresar nuevos principios de la ingeniería del software. Al contrario, estos intentan codificar el conocimiento, las expresiones y los principios ya existentes. (Larman, 2003)

2.6.1 Patrones GRASP

"Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresado en forma de patrones". (Larman, 1999)

Para el desarrollo de la presente investigación solo se utilizarán 4 patrones GRASP, estos son:

- **Experto:** consiste en asignar una responsabilidad al experto en información, en otras palabras, se asigna la responsabilidad a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Un ejemplo del uso del patrón en el componente Galería de imágenes es: el controlador (fwGaleriaDeImagenesCtrl)

tiene la responsabilidad de generar la galería dinámicamente a través del método $scope.aplicarIMG().

- **Alta cohesión:** Consiste en asignar una responsabilidad de modo para que la cohesión siga siendo alta. Este patrón se evidencia en las clases controladoras, las cuales tienen una serie de funcionalidades que se relacionan entre sí.
- **Bajo acoplamiento:** consiste en asignar una responsabilidad para mantener bajo acoplamiento. Este patrón se evidencia en que las clases controladoras de cada componente heredan de una única clase controladora generar.
- **Controlador:** consiste en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Es un objeto de interfaz no destinado al usuario que se encargue de manejar un evento del sistema. Este patrón se evidencia en las clases controladoras, ejemplo: fwGaleriaDeImagenes.controller.js

2.6.2 Patrones Gof

Gang of Four (GOF, en español significa pandilla de los cuatro) fue creado por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados por expertos diseñadores de software orientado a objetos. Estos patrones se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento. (Angel, 2011)

Para el desarrollo de la presente investigación se utilizarán los siguientes patrones GOF, estos son:

- ***Singleton*** (Instancia única): garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Se evidencia en la creación de los módulos.
- ***Observer*** (Observador): define una relación de uno a muchos entre objetos, de manera que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. Se evidencia al actualizar los valores de las variables de AngularJS.

2.7 Modelo de clases del diseño

"Los diagramas de clase del diseño describen específicamente las especificaciones de las clases de software y de las interfaces en una aplicación" (Larman, 2003).

A continuación solo se muestra el diagrama del componente Galería de imágenes, los restantes diagramas se pueden consultar en el Anexo 3: Diagrama de clase del diseño.

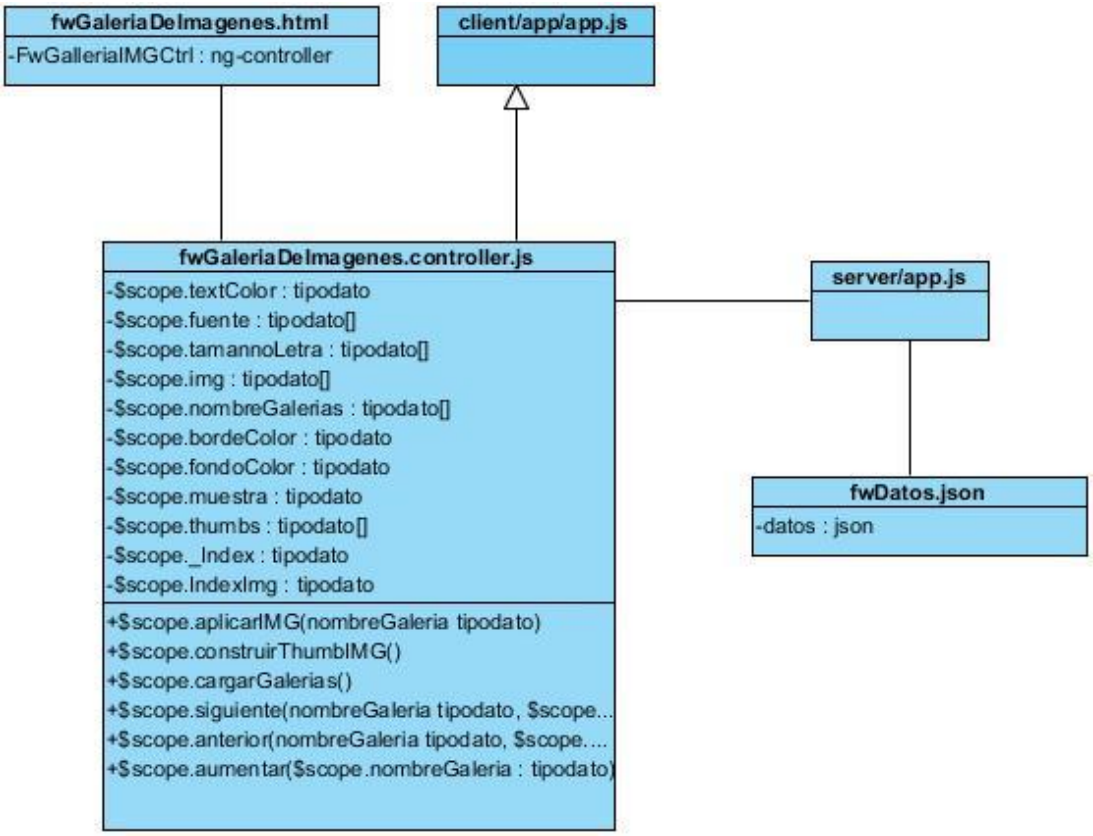


Figura 11: Diagrama de clase del diseño. Componente Galería de imágenes

2.8 Conclusiones del capítulo

Se confeccionó el modelo de dominio con el propósito de obtener una representación de los conceptos que intervienen en el desarrollo de los componentes. Los requisitos funcionales y no funcionales identificados permitieron determinar las principales funcionalidades que brindarán los componentes, con el propósito de darle cumplimiento al objetivo propuesto. A partir de los requisitos funcionales se realizaron las historias de usuario para determinar el tiempo que demora realizar cada requisito y la persona asignada para desarrollarlo. La arquitectura y los patrones de diseño seleccionados constituyeron una guía para la implementación de la propuesta de solución. Con el propósito de que los componentes cumplan con todas las funcionalidades identificadas y que sean implementados sin dificultad se realizaron los diagramas de clase del diseño.

Capítulo 3: Implementación y pruebas

3.1 Introducción

En el presente capítulo se describe la implementación de la propuesta de solución a través del diagrama de componente y de despliegue. Se definen los estándares de codificación a utilizar con el propósito de lograr una organización en el código de los componentes desarrollados. Además, se realizan las pruebas de software necesarias para obtener un producto de calidad y con las funcionalidades definidas, obteniendo los resultados de las mismas.

3.2 Implementación

La implementación consiste en la realización de una especificación técnica o algoritmos como un programa, componentes software u otro sistema de cómputo. Muchas implementaciones son dadas siguiendo una especificación o un estándar.

En la fase de implementación son generados una serie de artefactos ingenieriles como el diagrama de componentes.

3.2.1 Diagrama de componentes

El diagrama de componente permite visualizar la estructura de alto nivel del sistema y el comportamiento del servicio que estos componentes proporcionan y usan a través de interfaces. Los componentes pueden ser archivos, tablas, bibliotecas o documentos, estos son representados mediante un estereotipo. A continuación, se mencionan los estereotipos que conforman un diagrama de componente.

- **<<executable>>:** especifica un componente que se puede ejecutar en un nodo.
- **<<library>>:** especifica una librería de objetos estática o dinámica.
- **<<table>>:** especifica una tabla de una base de datos.
- **<<file>>:** especifica un fichero que puede contener código fuente o datos.
- **<<document>>:** especifica un documento.

A continuación, se muestra el diagrama de componente de la propuesta de solución:

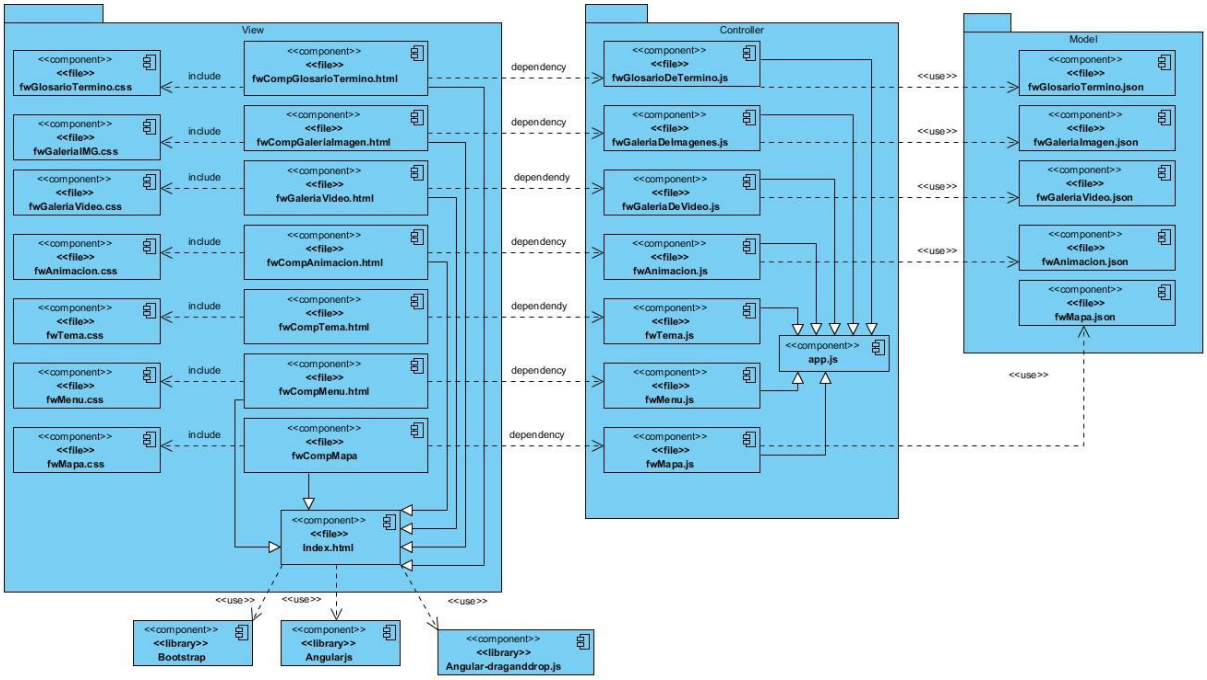


Figura 12: Diagrama de componentes

3.2.2 Estructura de la propuesta de solución

La estructura de la propuesta de solución emplea la arquitectura MVC, la cual divide el código fuente en tres capas como la vista, el modelo y el controlador. La utilización de esta arquitectura les permite a los proyectos de desarrollo mantener una organización del código. A continuación, se muestra la estructura de carpetas empleadas para la propuesta de solución.

La carpeta *client* es la vista del proyecto y dentro contiene una carpeta *app* generada por AngularJS. En la *app* se almacena una serie de subcarpetas que responde a cada uno de los componentes, ejemplo: *fwGaleriaDeImagenes*. Estas carpetas almacenan la vista y el controlador del componente que hace referencia, ejemplo: fwGaleriaDeImagenes.html y fwGaleriaDeImagenes.controller.js. La carpeta server contiene los archivos del servidor del proyecto. Además, contiene el modelo, este se accede atreves del directorio *fwMM_files/ fwMM_proyectos* el cual lista todas las multimedia. Dentro de la multimedia hay una carpeta *componentes*, esta contiene otra carpeta llamada *datosComponentes*. En esta carpeta se genera otra llamada fw seguido del nombre del componente, la cual va a contener otra carpeta con el identificador del componente que se encarga de almacenar los datos.

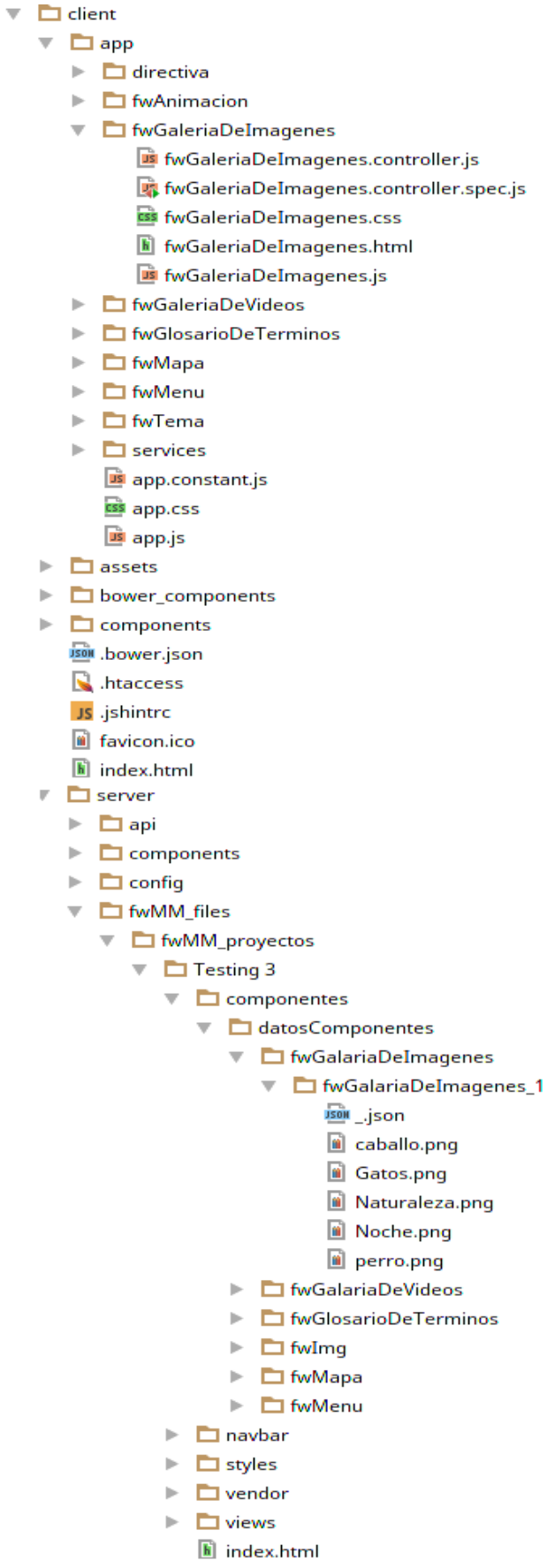


Figura 13: Estructura de carpetas de la propuesta de solución

3.3 Estilos y estándares de codificación

Los programadores que utilicen el lenguaje JavaScript suelen emplear una serie de reglas de estilo a la hora de escribir el código. Estas reglas las definen los programadores expertos o empresas como Google o Microsoft. No se consideran de obligado cumplimiento porque tan solo son recomendaciones que buscan que el código sea más fácil de leer, entender y que siga unos estándares. Para la implementación de la propuesta de solución se decide utilizar los siguientes estándares de codificación:

El estilo CamelCase indica que un conjunto de palabras se escribirán utilizando solo letras mayúsculas y minúsculas para diferenciarlas entre ellas. Cuenta con dos clases: UpperCamelCase y LowerCamelCase. UpperCamelCase consiste en que la primera letra de todas las palabras comiencen con mayúscula, ejemplo: GrupoManejadorBean. Mientras que LowerCamelCase define que la primera letra siempre empieza con minúscula, ejemplo: manejarOperacionSuma. Para la propuesta de solución se utilizará el LowerCamelCase. (Jonathan Nieto, 2011)

El estilo idiomatic.js indica como un programador JavaScript debe escribir el código. Entre los principales elementos que plantea son: los espacios en blanco, nunca se deben mezclar los espacios y las tabulaciones, los paréntesis, llaves, fines de línea, las declaraciones y las funciones. (GitHud, 2016)

3.4 Diseño de casos de prueba

El diseño de casos de prueba permite diseñar las entradas y las salidas esperadas para probar el sistema. El objetivo que persigue este proceso es crear un conjunto de casos de prueba que sean efectivos para descubrir defectos en los programas y muestren que el sistema satisface sus requerimientos. (Sommerville, 2005)

A continuación se muestran los Diseño de casos de prueba de solo 4 de los requisitos del componente Galería de imágenes, los demás se encuentran en el Anexo 6: Diseño de casos de prueba.

Tabla 6.Diseño de casos de prueba. Incluir descripción de la imagen

Descripción general: permitir incluir descripción de la imagen.				
Condiciones de ejecución: para incluir la descripción de la imagen hay que: -cargar la imagen.				
Nombre de la sección: SC1Incluir descripción de la imagen				
Escenarios	Descripción	Descripción	Respuesta del sistema	Flujo central
EC1.1 Incluir la descripción de	Escribe la descripción de la		El sistema brinda la posibilidad de	Galería de Imágenes/ Crear galería/

la imagen.	imagen.		escribir la descripción de la imagen en un campo de texto.	Imagen/ Descripción de la imagen
EC1.2 Opción de guardar la descripción	Selecciona la opción guardar, permitiendo guardar la descripción.		Guarda los datos de la descripción.	Galería de Imágenes/ Crear galería/ Guardar

Descripción de las variables

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Descripción	Campo de texto	No	El campo solo admite caracteres alfabéticos.

Tabla 7. Diseño de casos de prueba. Incluir título de la imagen

Descripción general: permitir incluir el título de la imagen.				
Condiciones de ejecución: para incluir el título de la imagen hay que: -cargar la imagen.				
Nombre de la sección: SC Incluir el título de la imagen				
Escenarios	Descripción	Título de la imagen	Respuesta del sistema	Flujo central
EC2.1 Incluir el título de la imagen.	Escribe el título de la imagen.		El sistema le debe permitir escribir el título de la imagen en un campo de texto.	Galería de Imágenes/ Crear galería/ Imagen/ Título de la imagen
EC2.2 Opción de guardar el título de la imagen.	Selecciona la opción guardar, permitiendo guardar el título de la imagen.		Guarda los datos de la descripción.	Galería de Imágenes/ Crear galería/ Guardar

Descripción de las variables

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Título de la imagen	Campo de texto	No	El campo solo admite caracteres alfabéticos.

Tabla 8. Diseño de casos de prueba. Configurar posición de los thumbnails

Descripción general: permitir configurar la posición de los thumbnails.							
Condiciones de ejecución: para configurar la posición de los thumbnails hay que: -cargar galería.							
Nombre de la sección: SC3 Configurar posición de los thumbnails.							
Escenarios	Descripción	right	left	bottom	top	Respuesta del sistema	Flujo central
EC3.1 Configurar posición de los thumbnails.	Seleccionar la opción que el usuario desee ya sea Right, Left, Bottom o Top.					El sistema le debe permitir seleccionar la posición de los thumbnails.	Galería de Imágenes/ Personalización/ Opciones/ /Posición de los thumbnails
EC3.2 Selecciona la opción Aplicar cambios	Selecciona la opción Aplicar cambios, permitiendo guardar la posición de los thumbnails.					Guarda la posición de los thumbnails.	Galería de Imágenes / Personalización/ Opciones/Aplicar cambios

Descripción de las variables

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Right	Campo de selección	No	Si se selecciona esta opción las demás no podrán ser seleccionadas.
2	Left	Campo de selección	No	Si se selecciona esta opción las demás no podrán ser seleccionadas.
3	Bottom	Campo de selección	No	Si se selecciona esta opción las demás no podrán ser seleccionadas.
4	Top	Campo de selección	No	Si se selecciona esta opción las demás no podrán ser seleccionadas.

Tabla 9.Diseño de casos de prueba. Incluir cantidad de columnas de los thumbnails

Descripción general: permitir incluir cantidad de columnas de los thumbnails.				
Condiciones de ejecución: para incluir la cantidad de columnas de los thumbnails hay que: -cargar galería.				
Nombre de la sección: SC4 Incluir cantidad de columnas de los thumbnails.				
Escenarios	Descripción	Columnas	Respuesta del sistema	Flujo central
EC4.1 Incluir cantidad de columnas de los thumbnails.	Permite seleccionar la cantidad de columnas de los thumbnails.		El sistema le debe permitir seleccionar la cantidad de columnas de los thumbnails.	Galería de Imágenes Personalización/ Opciones/ Cantidad de thumbnails
EC4.2 Selecciona la opción Aplicar cambios	Selecciona la opción Aplicar cambios, permitiendo guardar la cantidad de columnas de los thumbnails.		Guarda la cantidad de columnas de los thumbnails.	Galería de Imágenes / Personalización/ Opciones/Aplicar cambios

Descripción de las variables

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Columnas	Campo de selección	No	El campo solo admite caracteres numéricos.

3.5 Pruebas de software

Las pruebas de software son un elemento fundamental para la garantía de calidad del software. Además, representan una revisión final de las especificaciones, del diseño y de la codificación. (Pressman, 2005)

3.5.1 Pruebas unitarias

Las pruebas unitarias consisten en identificar y corregir problemas en una parte del código de una aplicación. El objetivo de esta prueba es evaluar de manera individual la correcta operación de cada módulo de software. (Jorge R. Aguilar Cisneros, 2015)

En el Generador de proyectos de AngularJS se utiliza Karma para configurar un proyecto de prueba preparada con todas las funcionalidades. Debido al empleo de la tecnología AngularJS en la presente investigación se decide utilizar la herramienta Karma.

El objetivo principal de **Karma** es ofrecer un entorno de pruebas productivas para los desarrolladores. Con el entorno no tienen que establecer las configuraciones, este ofrece un espacio donde los desarrolladores pueden simplemente escribir el código y obtener información inmediata de sus pruebas. (Karma, 2016)

En la figura 14 se muestra las pruebas unitarias realizadas al componente galería de imágenes. Las pruebas ejecutadas a los demás componentes se pueden consultar en el Anexo 7: Pruebas unitarias.

```
'use strict';
describe('Controller: FwGaleriaDeImagenesCtrl', function () {
  beforeEach(module('mdmMeanApp'));
  var FwGaleriaDeImagenesCtrl, scope;
  beforeEach(inject(function ($controller, $rootScope) {
    scope = $rootScope.$new();
    FwGaleriaDeImagenesCtrl = $controller('FwGaleriaDeImagenesCtrl', {
      $scope: scope
    });
    scope.imagenes = [
      {
        'titulo': "Amor",
        'descripcion': 'sentimiento de afecto universal que se tiene hacia una persona, animal o cosa',
        'dirImg': 'Mi Primera Multimedia/componentes/galeria_Img/datos/amor.jpg'
      },
      {
        'titulo': "Amigos",
        'descripcion': 'persona con quien se mantiene una amistad. Una amistad es una relación afectiva entre dos personas, construida
        'dirImg': 'Mi Primera Multimedia/componentes/galeria_Img/datos/amistad.jpg'
      },
      {
        'titulo': "Felicidad",
        'descripcion': 'estado emocional de una persona feliz; es la sensación de bienestar y realización que experimentamos cuando al
        'dirImg': 'Mi Primera Multimedia/componentes/galeria_Img/datos/feliz.jpg'
      }
    ];
  }));
  it('Cantidad de imagenes', function () {
    expect(scope.imagenes.length).toEqual(3);
  });
  it('Generar galeria de imagenes', function () {
    expect(scope.aplicarIMG).toBeDefined();
  });
  it('Comprobar expresion regular en el nombre de la galeria', function () {
    expect("Galeria de Mi multimedia").toMatch('^[a-zA-Z]+(\s*[a-zA-Z]*)*[a-zA-Z]+$');
  });
});
```

Figura 14: Pruebas unitaria del componente Galería de imágenes

Resultados de las pruebas unitarias

Las pruebas unitarias fueron realizadas a todos los componentes personalizables, los cuales conforman la propuesta de solución. Estas fueron ejecutadas satisfactoriamente obteniendo resultados favorables. En la Figura 15 se muestran los resultados obtenidos en las pruebas unitarias de los componentes.

```
Controller: FwAnimacionCtrl
  ✓ Método principal para animar
  ✓ Variable de animaciones no null
  ✓ Variable de estilos no null

Controller: FwGaleriaDeImagenesCtrl
  ✓ Cantidad de imagenes
  ✓ Generar galeria de imagenes
  ✓ Comprobar expresion regular en el nombre de la galeria

Controller: FwGaleriaDeVideosCtrl
  ✓ Cantidad de videos
  ✓ Generar galeria videos
  ✓ Comprobar expresion regular del titulo la galeria

Controller: FwGlosarioDeTerminosCtrl
  ✓ Adicionar letra
  ✓ Guardar palabra y descripcion y validar los mismos
  ✓ Datos no nulos
  ✓ Validar expresiones regulares

Controller: FwMapaCtrl
  ✓ Adicionar y actualizar punto
  ✓ Datos no nulos
  ✓ Cantidad de datos

Controller: FwMenuCtrl
  ✓ Guardar datos
  ✓ Eliminar datos
  ✓ Guardar enlace
  ✓ Guardar desplegable

 Controller: FwTemaCtrl
   ✓ Comprobar sistema Gird
   ✓ Verificar si scope.tomarColumnas esta definido

Chromium 48.0.2564 (Ubuntu 0.0.0): Executed 22 of 22 SUCCESS (0.227 secs / 0.178 secs)
TOTAL: 22 SUCCESS
```

Figura 15: Resultados de las pruebas unitaria

3.5.2 Pruebas de integración

La integración del sistema consiste en identificar grupos de componentes que proporcionan alguna funcionalidad del sistema e integrar estos añadiendo código para hacer que funcionen conjuntamente. Los componentes a integrar pueden ser: componentes comerciales, componentes reutilizables que han sido adaptados a un sistema particular, o componentes nuevos desarrollados. (Sommerville, 2005)

Las pruebas de integración se ocupan de probar las interfaces entre los componentes, las interacciones con distintas partes de un mismo sistema, como el sistema operativo, el sistema de archivos y el hardware y las interfaces entre varios sistemas. (Thomas Müller, 2010)

Puede existir más de un nivel de pruebas de integración y puede realizarse en objetos de prueba de distintos tamaños, como se indica a continuación:

- Las pruebas de integración de componentes se ocupan de probar las interacciones entre los componentes del software y se realizan a continuación de las pruebas de componente.
- Las pruebas de integración de sistema se ocupan de probar las interacciones entre los distintos sistemas o entre el hardware y el software. Estas pueden realizarse a continuación de las pruebas de sistema.

Las pruebas de integración realizadas en la presente investigación fue la de componentes, para probar la interacción de los componentes personalizables con el entorno de desarrollo.

Resultados de las pruebas de integración

En la presente investigación se realizaron las pruebas de integración a los componentes personalizables. En las pruebas se obtuvo como resultado para una primera iteración una no conformidad, la cual fue resuelta. En una segunda iteración se obtuvo resultados favorables, integrándose los componentes: Galería de imágenes, Galería de videos, Glosario de términos, Tema, Mapa y Menú, con el entorno de desarrollo. A continuación se muestra la imagen evidenciándose la integración.

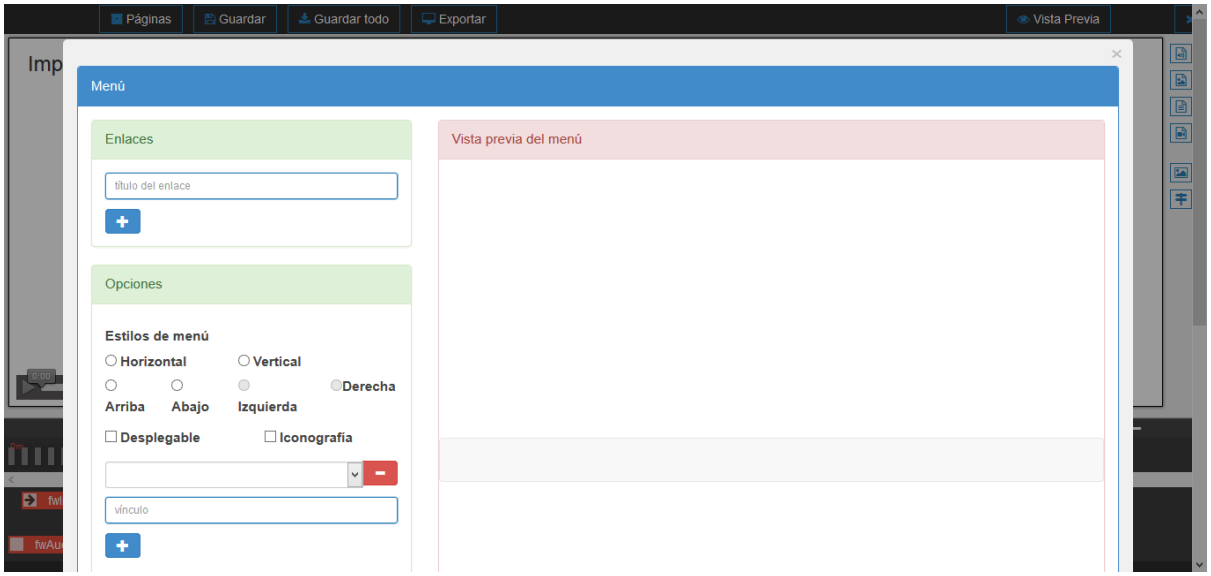


Figura 16: Resultado de la prueba de integración. Componente Galería de imágenes

3.5.3 Pruebas de aceptación

Las pruebas de aceptación en ocasiones son responsabilidad de los clientes o usuarios de un sistema, a pesar de que pueden participar otras partes interesadas. Estas pruebas consisten en evaluar la buena disposición de un sistema para su despliegue y uso. (Thomas Müller, 2010)

Las pruebas de aceptación pueden darse en distintos momentos del ciclo de vida como:

- Un producto de software puede ser objeto de pruebas de aceptación una vez instalado o integrado.

- Las pruebas de aceptación de la usabilidad de un componente pueden realizarse durante las pruebas de componente.
- Las pruebas de aceptación de una nueva mejora funcional pueden realizarse antes de las pruebas de sistema.

Resultados de las pruebas de aceptación

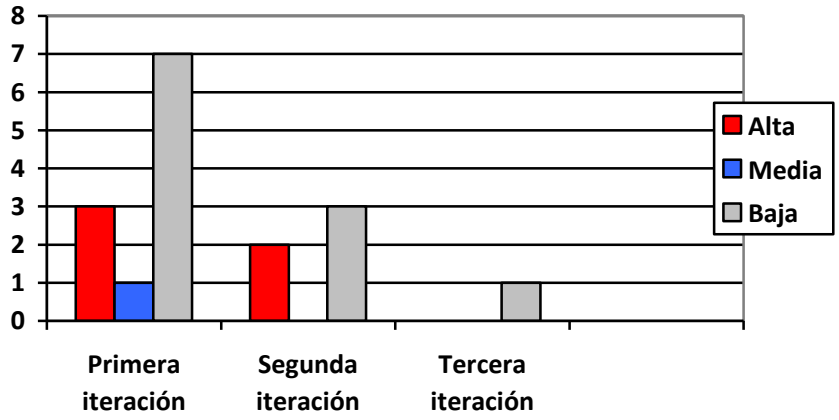
Las pruebas de aceptación fueron realizadas a todos los componentes personalizables que conforman la propuesta de solución. Para ello se efectuaron tres iteraciones, en las cuales se detectaron 17 no conformidades; de estas 5 se clasificaron de complejidad alta, 1 de complejidad media y 11 de complejidad baja.

Las no conformidades fueron clasificadas según su complejidad por lo que se definen de la siguiente forma:

- Alta son los errores en el código o errores de funcionalidad.
- Media son los errores de ortografía o de validación.
- Baja son los errores de interfaz.

A continuación se expone un gráfico que muestra la relación entre las no conformidades detectadas de complejidad alta, media o baja.

Tabla 10. No conformidades



Una vez concluida cada iteración de pruebas se analizaron por parte del equipo de desarrollo las no conformidades encontradas y se determinaron las que constituían fallas del sistema o prestaban variaciones en las Historias de usuario.

3.6 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas existentes entre los componentes hardware y software en el sistema final. Es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes de software como

son los procesos y objetos que se ejecutan en ellos. (Ivar Jacobson, 2000) A continuación, se muestra el diagrama de despliegue de la propuesta de solución.

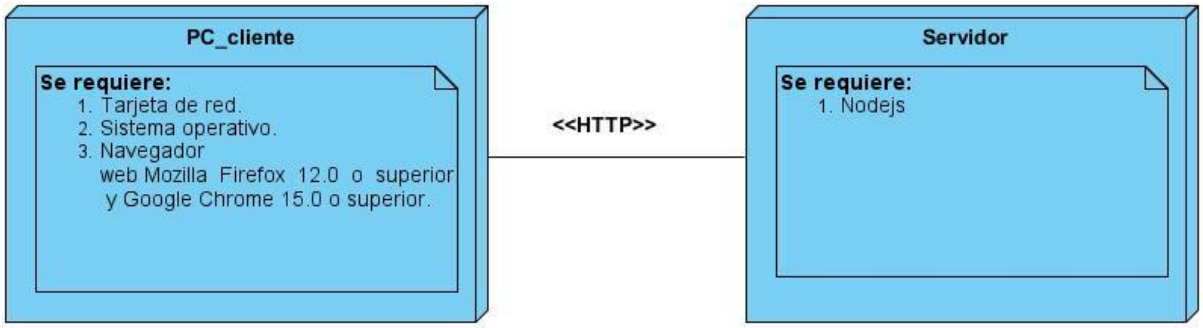


Figura 17: Diagrama de despliegue

3.7 Conclusiones del capítulo

Se generaron los diagramas de componentes y despliegue para entender la relación de dependencia y uso de los componentes, así como la relación entre los componentes hardware y software. Los estilos y estándares de codificación CamelCase e idiomatic.js permitieron organizar el código de la propuesta de solución. Los diseños de casos de pruebas permitieron identificar los errores del programa. Además, se realizaron un total de 22 pruebas unitarias a la aplicación con la herramienta Karma de las cuales ninguna fallida en la última ejecución, lo que demuestra la calidad del sistema. También se aplicaron las pruebas de integración donde los componentes fueron integrados en el entorno de desarrollo obteniendo resultados satisfactorios. Las pruebas de aceptación realizadas a los componentes permitieron detectar algunas no conformidades, las cuales fueron corregidas para mejorar la calidad y funcionalidad de los componentes.

Conclusiones generales

El desarrollo de los componentes personalizables en conjunto con el análisis de los resultados obtenidos, posibilitaron obtener las siguientes conclusiones:

- El estudio del estado del arte permitió identificar los conceptos relacionados con el objeto de estudio.
- El análisis de las soluciones similares posibilitó determinar las funcionalidades bases para el desarrollo de los componentes.
- Los artefactos ingenieriles generados permitieron describir la propuesta de solución.
- Para darle cumplimiento al objetivo general, se desarrollaron siete componentes personalizables utilizando las tecnologías y herramientas: AngularJS, Bootstrap, Nodejs, WebStorm, Yeoman, Bower, Grunt, Jasmine, Express y NPM.
- Los componentes personalizables desarrollados permitieron mejorar el proceso de creación de multimedia en el centro FORTES.
- Se aplicaron pruebas unitarias, de integración y aceptación las cuales permitieron determinar y erradicar las no conformidades encontradas.

Recomendaciones

A partir del trabajo realizado se recomienda al proyecto Marco de trabajo para el desarrollo de multimedia con tecnologías libres:

- Agregarle la iconografía al componente Mapa.
- Dibujar la trayectoria de la animación.
- Integrar el componente Animación al Entorno de desarrollo.
- Permitir que el mapa se mueva dentro del área de trabajo.

Referencias bibliográficas

1. **Alvarez, Alberto Basalo y Miguel Angel. 2014.** Qué es AngularJS. [En línea] 28 de agosto de 2014. [Citado el: 15 de enero de 2016.] http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html.

2. **Angel. 2011.** Geed the pianet. [En línea] mayo de 2011. [Citado el: 1 de marzo de 2016.] http://geektheplanet.net/5462/patrones-gof.xhtml.

3. **Aponte, Ángela María Valbuena. 2012.** *Guía comparativa de frameworks para los lenguajes HTML 5, CSS y Javascript para el desarrollo de aplicaciones web.* Universidad Tecnológica de Pereira : s.n., 2012. Tesis.

4. **Belloch, Consuelo. 2012.** APLICACIONES MULTIMEDIA. [En línea] 2012. [Citado el: 2 de diciembre de 2015.] http://www.uv.es/bellochc/logopedia/NRTLogo4.pdf.

5. **Bootstrap. 2016.** Bootstrap. [En línea] 2016. [Citado el: 4 de febrero de 2016.] http://getbootstrap.com/.

6. **Bower. 2012.** Bower. [En línea] 2012. [Citado el: 17 de mayo de 2016.] http://bower.io/.

7. **Cachero, Cristina. 2010.** *Personalización de Aplicaciones en OO-H .* 2010.

8. **CSS3. 2016.** CSS3. [En línea] 25 de enero de 2016. [Citado el: 4 de febrero de 2016.] https://developer.mozilla.org/es/docs/Web/CSS/CSS3.

9. **CUEVAS, NIELS HENRYK ARANDA. 2014.** [En línea] 2014 de junio de 2014. [Citado el: 7 de febrero de 2016.] http://es.slideshare.net/GiancarloAguilarChe/componentes-y-libreras-tpicos-avanzados-de-programacin.

10. **Datos, Grupo de Ingeniería del Software y Bases de. 2013.** *Documentación de requisitos de cliente: Historias de usuario.* Sevilla : s.n., 2013.

11. **Express. 2016.** Express. [En línea] 2016. [Citado el: 9 de junio de 2016.] http://expressjs.com/es/.

12. **Expto. Oscar Zalazar, Expto. Pedro L. Alfonzo, Lic. Yanina Medina, Osvaldo P. Quintana, Lic. Lucía Salazar. 2012.** *Taller de programación 1.* 2012.

13. **Florentino, Marvin. 2016.** Mozilla Developer Network (MDN). [En línea] 24 de febrero de 2016. [Citado el: 1 de marzo de 2016.] https://developer.mozilla.org/es/docs/Web/HTML.

14. **Flores, Araceli Soledad Domínguez. 2013.** Unidad 1: UML y el proceso unificado. *Desarrollo e implementación de Sistemas de Información.* 2013.

15. **Fuentes, Inés Meriño. 2012.** *Arquitectura de software.* Universidad del Magdalena : s.n., 2012.

16. **Gelderen, María Marta van. 2011.** *Multimedia educativa.* Ciudad Autónoma de Buenos Aires : s.n., 2011.

17. **GitHud. 2016.** GitHud. *Principios para escribir JavaScript consistente e idiomático.* [En línea] 2016. [Citado el: 28 de abril de 2016.] https://github.com/rwaldron/idiomatic.js/tree/master/translations/es_ES.

18. **González, Rolando Alfredo Hernández León y Sayda Coello. 2011.** *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011. ISBN 978-959-16.

19. **Grunt. 2016.** Grunt. [En línea] 2016. [Citado el: 17 de mayo de 2016.] http://gruntjs.com/getting-started.

20. **Hernández, M.C. Norma Ramírez, López, M.S.I Graciela Lara y Chávez, M.A.S.I. Salomón Eduardo Ibarra. 2014.** *Análisis, diseño y programación de sistemas.* 2014. ISBN 970764803-1.

21. **Informáticas, Portal de la Universidad de las Ciencias. 2012.** Portal de la Universidad de las Ciencias Informáticas. [En línea] 2012. [Citado el: 26 de noviembre de 2015.] http://www.uci.cu/?q=mision.

22. **Ivar Jacobson, Grady Booch, James Rumbaugh. 2000.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000. ISBN: 74-7829-036-2.

23. **Jasmine. 2016.** Jasmine. [En línea] 2016. [Citado el: 9 de junio de 2016.] http://jasmine.github.io/2.4/introduction.html.

24. **JavaScript Engines, o motores JavaScript. 2015.** JavaScript Engines, o motores JavaScript. [En línea] 2015. [Citado el: 4 de mayo de 2016.] https://moodle2015-16.ua.es/moodle/pluginfile.php/26075/mod_resource/content/5/page_08.htm.

25. **Jonathan Nieto, Marco Martínez. 2011.** *Sistema Wed Ayni. Estandares de programación. .* 2011.

26. **Jorge R. Aguilar Cisneros, Carlos Alberto Fernández-y-Fernández. 2015.** *Especificación de Requerimientos para el Desarrollo de Software Automotriz en México.* México : s.n., 2015. ISSN 2314-2642.

27. **Karma. 2016.** Karma. [En línea] 2016. [Citado el: 27 de abril de 2016.] http://karma-runner.github.io/0.8/index.html.

28. **Labrada, Sonia Morejón. 2011.** Cuadernos de educación y desarrollo. *El software educativo un medio de enseñanza eficiente.* [En línea] julio de 2011. [Citado el: 26 de noviembre de 2015.] http://www.eumed.net/rev/ced/29/sml.htm.

29. **Larman, Craig. 1999.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : s.n., 1999. ISBN 970-17-0261-1.

30. **Layoutit. 2015.** Layoutit. [En línea] 2015. [Citado el: 17 de mayo de 2016.] http://www.layoutit.com/.

31. **Lennon, Joe. 2011.** Cree sitios Web modernos usando HTML5 y CSS3. *Cree sitios Web modernos usando HTML5 y CSS3.* [En línea] 14 de junio de 2011. [Citado el: 4 de febrero de 2016.] https://www.ibm.com/developerworks/ssa/web/tutorials/wa-html5/#ibm-pcon.

32. **Microsoft. 2015.** Microsoft. [En línea] 2015. [Citado el: 5 de abril de 2016.] https://msdn.microsoft.com/es-es/library/dd409390.aspx.

33. **Nodejs. 2015.** Nodejs. [En línea] 6 de febrero de 2015. [Citado el: 5 de febrero de 2016.] https://nodejs.org/en/blog/release/v0.12.0/.

34. **NPM. 2016.** NPM. [En línea] 13 de marzo de 2016. [Citado el: 9 de junio de 2016.] https://docs.npmjs.com/getting-started/what-is-npm.

35. **Paradigm, Visual. 2015.** Visual Paradigm. [En línea] 2015. [Citado el: 13 de Enero de 2016.] http://www.visual-paradigm.com/aboutus/newsreleases/vpuml80.jsp.

36. **Pérez, Javier Eguíluz. 2009.** *Introducción a CSS.* 2009.

37. **PMBOK. 2008.** *Guía de los fundamentos para la dirección de proyectos (Guía del PMBOK). Cuarta Edición.* 2008. ISBN: 978-1-933890-72-2.

38. **Portillo, M. del Pilar del Saz. 2014.** *Tutorial Patrón MVC.* 2014.

39. **Pressman, Roger S. 2005.** *Ingeniería de software. Un enfoque práctico. 5ta Edición.* New York : s.n., 2005. ISBN 978-0-07-337597-7.

40. **Real, Víctor Perlacia y Carballo, Yeidy Martínez. 2015.** *Componentes básicos para el marco de trabajo de desarrollo de multimedia con tecnologías libres.* La Habana : s.n., 2015.

41. **Reyes, Rebeca Abigail González, Eugenio, Aarón Augusto Pérez, González, Dulce Fabiola Ramos, Chávez, Mayte Rivera. 2014.** *Estereotipos e Interfaces.* Tijuana : s.n., 2014.

42. **Ricardo Barrera Urieta, María Elizabeth García Mayo, Joaquín Herrera Nova, Adilene Ríos Urbina. 2014.** *Plataformas de Desarrollo de Aplicaciones Móviles.* Costa Grande : s.n., 2014.

43. **Rodríguez, Msc. Dayra Iris Hechavarría. 2012.** Gestión de Requisitos. *Gestión de Requisitos.* [En línea] 2012. [Citado el: 2016 de febrero de 25.] http://www.monografias.com/trabajos92/gestion-requisitos/gestion-requisitos.shtml#queesunrea.

44. **Sánchez, Tamara Rodríguez. 2014.** *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : s.n., 2014.

45. **Scholz, Florian. 2015.** Mozilla Developer Network (MDN). [En línea] 26 de abril de 2015. [Citado el: 1 de marzo de 2016.] https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript.

46. **Sommerville, Ian. 2005.** *Ingeniería de software, 7ma edición.* Madrid : Pearson Educación, S.A., 2005. ISBN: 84-7829-074-5.

47. **Sommerville, Ian. 2007.** *Ingeniería de Software. 8va Edición.* 2007. ISBN 10: 0-321-31379-8.

48. **svBuilder-Pro. 2016.** svBuilder-Pro. [En línea] 2016. [Citado el: 22 de febrero de 2016.] https://www.simpleviewer.net/simpleviewer/pro/support/svbuilderpro/.

49. **Thomas Müller, Armin Beer, Martin Klonk, Rahul Verma. 2010.** *Probador Certificado. Programa de estudio de nivel básico.* Ciudad de la Habana : s.n., 2010.

50. **Tutorialspoint. 2016.** Angularjs- MVC Architecture. [En línea] 2016. [Citado el: 16 de marzo de 2016.] http://www.tutorialspoint.com/angularjs/angularjs_mvc_architecture.htm.

51. **WebStorm. 2016.** JetBrains. *WebStorm.* [En línea] 2016. [Citado el: febrero de 8 de 2016.] https://www.jetbrains.com/webstorm/features/ide-features.html.

52. **Yenisleidy Fernández Romero, Yanette Díaz González. 2012.** Revista Telem@tica. *Revista Telem@tica.* [En línea] enero-abril de 2012. [Citado el: 25 de febrero de 2016.] http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/15/10. ISSN 1729-3804.

53. **Yeoman. 2016.** Yeoman. [En línea] 2016. [Citado el: 17 de mayo de 2016.] http://yeoman.io/.