Universidad de las Ciencias Informáticas

Facultad 4

"Componentes básicos 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:
Víctor Perlacia Real
Yeidy Martínez Carballo
Tutores:
Ing. Lizandra Hernández Hernández
Msc. Rosalba Carralero Medina

La Habana, Junio 2015

"Año 56 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):

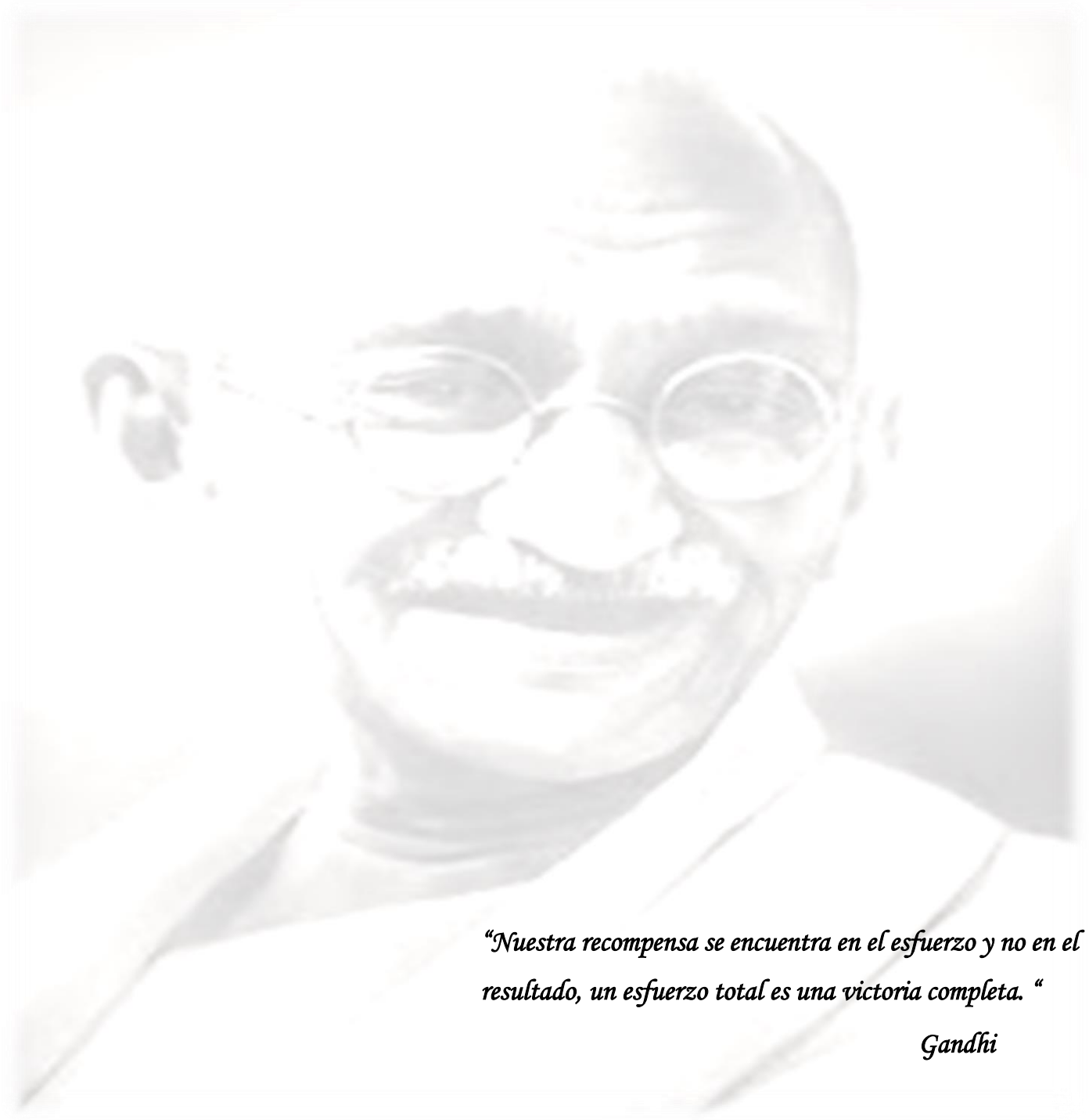
_____ _____

Víctor Perlacia Real Yeidy Martínez Carballo

Tutores:

_____ _____

Ing. Lizandra Hernández Hernández Msc. Rosalba Carralero Medina



"Nuestra recompensa se encuentra en el esfuerzo y no en el resultado, un esfuerzo total es una victoria completa. "

Gandhi

De Yeidy:

Con todo el amor que nace dentro de mi corazón, dedico esta tesis a mi mami, mi gorda linda, que me ha dado todo lo que tengo, sin ella este sueño no hubiera sido posible. A mi tatongo, que me alumbró la vida desde que supe que estaba formándose en la pancita de mi mami. A mi abuelito Ivo, que siempre confió en mí, aquí está abue, lo logré. A mi viejuco lindo, que soñaba con este día, te amo pa, espero estés orgulloso de mí.

De Víctor:

Dedico la tesis a mis padres.

De Yeidy:

Desde pequeña soñaba con entrar a la universidad y convertirme en el orgullo de mi familia. Cuando lo logré, planificaba en mi cabeza qué decir el día de mi tesis. Hoy estoy aquí, y confieso que se me hace muy difícil agradecer a todos los que en estos 5 años confiaron en mí, aun cuando resultaba descabellado hacerlo…Primeramente, quiero agradecer a mi gorda linda, la manzanita de mi ojos, que ha sido todo en mi vida: mi mamá, mi papá, mi mejor amiga y la hermana que no tuve, la que siempre ha estado ahí mostrándome lo bueno de la vida y ayudándome cuando me he equivocado. Gracias por estar mi linda!!! A mi abuelito, ese viejo resabioso que me amó mucho, aunque no siempre supo demostrarlo, que siempre confió en mí, que me celaba como algo muy de él que no quería que nadie tocara. Estoy convencida que donde quiera que esté, está muy orgulloso de lo que he conseguido, te amo abu. A mi viejuco, ese padre de corazón que la vida me regaló y que me quitó hace 2 años, gracias por haber existido en mi vida, por cuidarme y ayudarme siempre. Jamás te perdonaré que te hayas ido así sin avisar, sin verme graduada como siempre quisiste. Te adoro pa, y siempre estarás vivo en mi corazón. A mi tatongo, mi niño que ya es un hombre, te amo tato. A mi abu Nora, mi vieja linda, gracias por estar a mi lado desde que nací, gracias por aguantar mis resabios, jeje sin ti esto no sería posible. A mi abuela Cora, por ocuparse de mí, por quererme y apoyarme siempre, gracias mami. A mi tito lindo, por estar pendiente de mí, por sufrir con mis problemas y hacerme reír tanto con sus cosas, gracias tito, sabes que te adoro. A mi tía Hania, como no agradecerle lo que ha hecho por mí y por mi familia en estos 7 años, y sobre todo por darme esos primos bellos, a los cuales amo con todo mi corazón. A mi papá, que a pesar de no estar siempre, sé que me quiere y te agradezco que estés hoy a mi lado. A mi hermano Bárbaro, que me ha demostrado la fuerza de la sangre, te quiero mucho mi hermanito. A mi cielín, que me ha enseñado que hay que ser valiente cuando se trata de amor, gracias por estar!!! No te apartes nunca mi amor, sé que a pesar de todo, vale la pena. A mi suegra linda por ocuparse tanto de mí, y estar pendiente de cada detalle de mi tesis. A mimi, mami y toda la familia de mi cielín, que ya la siento mía y que sé que me quieren mucho. A las pelúas, mis amigas Yul y Celita y al

cabezón Deivid, sin ustedes nada hubiera sido posible. Gracias por siempre estar, mostrándome lo lindo de una amistad verdadera. A los amigos de siempre: La figura, Santona, Viki, el Fide, Omar, gracias por todo el tiempo compartido, los quiero mucho. A Ney y Niurki, como no mencionarlas… Gracias 1000 por todo. A mi profe, amiga y tutora, Rosalba, por ayudarme tanto desde tercer año, gracias por la paciencia, los consejos y el cariño demostrado, sin duda se coló en mi corazón. A Roa, gracias amigo por las horas dedicadas desde segundo año con AC, jeje, este triunfo es tuyo también. A Eddito, que no me puede ver graduada, pero formó parte invaluable de este sueño, Gracias primote. A mi despistado compañero de tesis, creo que sin él me hubiera vuelto loca, Gracias Víctor. A los profes que siempre confiaron en mí y me ayudaron cuando todo parecía perdido: Rafael, Maritza e Irán. A todos gracias, sin ustedes nada hubiera sido posible, esta tesis también es suya.

De Víctor:

Ante todo quiero darle Gracias a Dios quien me ha ayudado a llegar hasta donde estoy, quien me enseña, me guía y sobre todo me ama como nadie nunca me amó ni me amará jamás. Le doy gracias a mi esposa, la cual amo mucho y ha sido una ayuda idónea en todo este tiempo, al igual que mi amigo Yoandri, los cuales me han ayudado en cosas más grandes y mejores que la tesis. Agradezco a mis padres pues son un ejemplo a seguir, a Laurita y Reinaldo, a mis abuelos, a mi hermana y su esposo que me quieren mucho a Tía y a Tomasito, a Ariel y Alejandro, a Maidani y Leidis, Yoyita y López, Rafelito y Modesta, a todo el Familión y el personal del callejón de los guerra que es bastante grande. Le doy gracias a mi compañera de tesis Yeidy pues sin ella este trabajo no se habría realizado, a los tutores, por su ayuda, gracias a Roanny, José Antonio y Adrián que han sido como tutores para nosotros, le doy gracias al oponente y tribunal. A mis hermanos en la fe, a mi familia de la UCI, con los que he compartido todo este tiempo, mis compañeros de batalla, los quiero un montón, a todos los presentes y aun a los ausentes, gracias. Dios les Bendiga.

En la Universidad de las Ciencias Informáticas (UCI) el proceso de desarrollo de multimedia involucra el uso de herramientas privativas, lo que imposibilita la comercialización de los productos obtenidos. La presente investigación está dirigida a desarrollar un grupo de componentes básicos, que serán integrados a un marco de trabajo libre y multiplataforma, con el cual se logrará crear *software* multimedia en un menor tiempo, y con una mayor calidad. El desarrollo de la propuesta de solución fue guiado por la metodología de desarrollo AUP. Como lenguajes de programación del lado del cliente se utilizaron HTML5, JavaScript y CSS3, apoyados en los *frameworks* AngularJS, JQuery en su versión 2.0.3 y Twitter Bootstrap en su versión 3.3.4. Se utilizó el NetBeans IDE como Entorno de Desarrollo Integrado y Visual Paradigm como herramienta de modelado UML. Se implementaron los componentes: visor de imágenes, visor de videos, glosario de términos, menú, buscador y trabajo con mapas. Fueron realizadas las pruebas de *software* a la propuesta de solución, las cuales aseguran en gran medida la correcta implementación de las funcionalidades y un alto grado de satisfacción por parte del cliente.

Palabras claves: Componentes básicos, Marco de trabajo, Multimedia.

Índice

Índice de figuras

Índice de tablas

Introducción

El siglo XXI es conocido por los expertos como "el inicio de la era de la información". Es la etapa donde las Tecnologías de la Información y las Comunicaciones (TIC) han revolucionado la concepción tradicional de la salud, la economía y la educación.

La revolución cubana desde sus inicios en enero de 1959 se trazó la tarea de nacionalizar la educación en el país. En la actualidad, esta labor se ve materializada en los centros educacionales que existen a lo largo de la isla. Un ejemplo invaluable de los esfuerzos realizados por el gobierno cubano, lo constituye la Universidad de las Ciencias Informáticas (UCI). Como universidad creada bajo la concepción de fomentar el aprendizaje conjuntamente con la producción, es el pilar nacional en el desarrollo de *software*, dentro de los que se encuentra la multimedia.

El *software* multimedia hace referencia al uso combinado y coherente de medios como imágenes, sonido, textos, animaciones y videos, y permiten la interacción con el usuario. (Belloch, 2012) Las mejores tecnologías existentes en el mundo para llevar a cabo el proceso de desarrollo de multimedia y de *software* en general poseen una licencia privativa. Razón que motiva a la UCI a comenzar un proceso de migración hacia nuevas plataformas que garanticen la independencia tecnológica y la libertad de comercio de software con el mundo. El centro de Tecnologías para la Formación (FORTES) de la UCI es el encargado de producir *software* educativos, dentro de los cuales prevalece la multimedia. Dicho desarrollo es realizado con las herramientas que provee el paquete ADOBE CS6, que suponen una serie de limitantes para la creación de sistemas informáticos de este tipo. Entre los principales problemas que presenta se encuentran el impacto negativo que tiene en la economía del país el pago de la licencia para su uso. Además, se obtiene como resultado sistemas informáticos que no son multiplataforma, condición que no se ajusta al proceso de migración que se lleva a cabo en la universidad.

La principal deficiencia se encuentra en la necesidad que existe de una herramienta libre capaz de ofrecer un ambiente de desarrollo para la creación de multimedia. De ahí que FORTES se plantee la tarea de desarrollar componentes basados en tecnologías libres para ser integrados a un marco de trabajo que contribuya con el proceso de creación de *software* multimedia. "*Un marco de trabajo se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta"* (Valbuena, 2014). Entre los principales objetivos de un marco de trabajo

se encuentra reutilizar el código ya existente. Además, permite acelerar el proceso de desarrollo y promover las buenas prácticas de creación de *software* con el uso de patrones (Valbuena, 2014).

La situación problémica antes descrita, da paso al siguiente **problema a resolver:** ¿Cómo contribuir con la creación de un marco de trabajo para el proceso de desarrollo de multimedia con tecnologías libres?

Del problema anterior se deriva como **objeto de estudio** el proceso de desarrollo de multimedia. Determinando como **campo de acción** de la investigación**,** el desarrollo de los componentes básicos para la creación de multimedia con tecnologías libres.

El **objetivo general** de este trabajo es desarrollar los componentes básicos 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 existente sobre el tema.
- Realizar el análisis y diseño de los componentes básicos para el marco de trabajo de desarrollo de multimedia.
- Implementar los componentes básicos para el marco de trabajo de desarrollo de multimedia.
- Realizar pruebas para garantizar la funcionalidad de los componentes básicos.

Las **tareas de investigación** a realizar para dar cumplimiento a los objetivos específicos son:

- Elaboración del estado del arte de los principales elementos teóricos del tema a tratar.
- Análisis y diseño de los componentes básicos.
- Confección de la propuesta de solución.
- Realización de pruebas para detectar anomalías y no conformidades.

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

El **resultado** que se obtendrá con esta investigación será: un conjunto de componentes básicos que contribuyan con el proceso de creación de *software* multimedia desde un marco de trabajo basado en tecnologías libres.

"*El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. Se puede clasificar en teóricos y empíricos, los cuales están dialécticamente relacionados.*" (Hernández, Coello, 2002). Con el fin de desarrollar la presente investigación de manera óptima, se realizó un estudio de los **métodos científicos** existentes para seleccionar los que se adaptan al objetivo general y a las tareas de investigación previstas. A continuación se citan los métodos teóricos y empíricos seleccionados:

Métodos teóricos

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

Histórico-Lógico: se empleó en el estudio organizado de la evolución de las multimedia, con el objetivo de conocer cómo se ha comportado su desarrollo en Cuba, enfatizando en la UCI.

Métodos empíricos

Entrevista: se utilizó para identificar las necesidades reales de la línea de multimedia del centro FORTES y de esta manera definir el objetivo al que estará dirigida la propuesta de solución.

Para lograr una mejor comprensión de la presente investigación se definió la siguiente estructura capitular:

Capítulo 1: Fundamentación teórica.

En este capítulo se abordan los principales conceptos que fundamentan la presente investigación. Se realiza un análisis de las soluciones similares existentes en Cuba y se describe la metodología, las herramientas y tecnologías que serán empleadas para crear los componentes básicos para el marco de trabajo de desarrollo de multimedia dentro del centro FORTES.

Capítulo 2: Descripción de la propuesta de solución.

En este capítulo se expondrán los requisitos funcionales y no funcionales que tendrá que cumplir la propuesta de solución. Además, se generan los artefactos de ingeniería correspondientes a la metodología de desarrollo de *software* seleccionada.

Capítulo 3: Implementación y pruebas.

En este capítulo se presenta una descripción del proceso de implementación de la propuesta de solución. Además, se detallan las pruebas realizadas con el objetivo de lograr un resultado que cumpla con los requerimientos del cliente.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se definen los principales conceptos que serán abordados durante la investigación, sus características y aplicaciones en la actualidad. Además, se describe la metodología de desarrollo de *software*, las herramientas y tecnologías que serán utilizadas en la propuesta de solución.

1.2 Generalidades de las multimedia

En la actualidad al término multimedia se le ha dado disímiles enfoques. A continuación se muestran algunos de los conceptos que han sido analizados, con el objetivo de entender mejor qué es una multimedia:

- "Una aplicación multimedia, es un programa que integra y maneja distintos tipos de información multimedia (imágenes, videos, bases de datos, simulaciones, textos, audio,…)" (Nicado, 2014).
- En el ámbito de la informática, la expresión multimedia se utiliza para calificar y describir las cualidades de diversos sistemas, aplicaciones, documentos y productos que emplean una combinación de textos, gráficos, imágenes, vídeos, animaciones y sonidos. Además incluyen hipervínculos, para permitir a los usuarios desplazarse por la información de forma intuitiva e interactiva (Pinto, 2011).
- El término multimedia hace referencia al uso combinado de diferentes medios de comunicación: texto, imagen, sonido, animación y video. Los programas informáticos que utilizan de forma combinada y coherente con sus objetivos diferentes medios, y permiten la interacción con el usuario son aplicaciones multimedia interactivas (Belloch, 2012).

Las definiciones anteriormente expuestas contribuyeron a elaborar un concepto más generalizado de multimedia. Una multimedia es un sistema informático interactivo, que se basa en la combinación coherente de medios de información como: imágenes, videos, sonido, textos y animaciones. Tiene la misión de expresar un determinado conocimiento al usuario, de manera dinámica y fácil de entender.

Según Gallego y Alfonso (1995), la multimedia cuenta con cuatros características principales (Cabrero y Duarte, 2015):

1.2.1 Características principales de una multimedia

- **Interactividad:** se denomina interacción a la comunicación que existe entre la máquina y el usuario, la acción que puede producir un usuario en el ordenador y la respuesta que es brindada por la computadora.
- **Ramificación:** se entiende por ramificación a la capacidad que tienen los sistemas multimedia de responder de acuerdo a lo que solicite el usuario.
- **Transparencia:** el usuario debe estar centrado en el mensaje que quiere transmitir la multimedia y no en el medio que emplea para hacerlo.
- **Navegación:** la capacidad que poseen las multimedia para que el usuario pueda desplazarse por ellas, de manera no secuenciada linealmente.

1.2.2 Medios de información

El uso de los diferentes medios de información juega un papel protagónico dentro de cualquier programa informático, puesto que puede potenciar la memoria visual y auditiva, además de aumentar el nivel de aprendizaje del usuario. A continuación se brinda una breve descripción de estos medios (Belloch, 2012):

- **Texto:** su inclusión en las aplicaciones multimedia brinda entre otras ventajas, la de desarrollar la comprensión lectora y la fluidez verbal del usuario. El texto tiene como función principal favorecer la reflexión y profundización en los temas, potenciando el pensamiento de más alto nivel.
- **Sonido:** son incorporados en las aplicaciones multimedia principalmente 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. El uso de este medio en las multimedia es altamente relevante para potenciar el aprendizaje de idioma y en las aplicaciones multimedia cuya finalidad es la intervención en problemas de comunicación y lenguaje.
- **Iconográficos:** permiten la representación de palabras, conceptos, ideas, mediante imágenes y gráficos. Su carácter visual le brinda un carácter universal, por lo que resulta adecuado en aplicaciones que serán utilizadas por personas que hablan otros idiomas.
- **Imágenes estáticas:** ilustran y facilitan la información que se desea transmitir en una multimedia. Pueden ser ilustraciones, fotografías, representaciones gráficas o fotogramas.
- **Imágenes dinámicas (Video o Animaciones):** son un recurso de gran importancia, puesto que transmiten de forma visual secuencias completas de contenido, ilustrando una parte del contenido que tiene sentido propio. Ayudan a expresar un evento que es difícil de observar en forma real. La

animación permite un mayor control de las situaciones, mediante esquemas y figuraciones que la imagen real reflejada en los videos no posibilita.

- **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 son gestionados por una serie de componentes, razón por la cual la creación de *software* multimedia puede ser analizada como un proceso basado en componentes. Un **componente** se define como una pieza de *software* auto-contenido que ofrece unos servicios definidos en sus interfaces y que está preparada para integrarse en otras aplicaciones (Sánchez, Sicilia y García, 2012). El uso de componentes brinda un grupo de beneficios para el proceso de desarrollo de multimedia. Entre las principales ventajas que ofrece se encuentra la posibilidad de utilizarse en otras aplicaciones, la reducción de costos y el mejoramiento de la calidad.

Con el objetivo de determinar las características y funcionalidades que pueden servir como base para el desarrollo de la propuesta de solución, además de conocer los componentes que resultan básicos en las multimedia, se realizó un análisis de las multimedia desarrolladas en el Centro FORTES.

1.3 Análisis de soluciones similares

Colecciones Tipo

La multimedia Colecciones Tipo fue solicitada por el Acuario Nacional Cuba, con el objetivo de que los investigadores o especialistas interesados en el tema puedan estudiar las colecciones sin necesidad de interactuar directamente con las mismas. Se pretende con el producto desarrollado mostrar a los usuarios la información coleccionada por los especialistas del Acuario Nacional Cuba, organizada por reino, phylum, orden, clase, familia, género y especie. La multimedia desarrollada está compuesta por cinco pantallas principales: la pantalla de inicio, pantalla de contenido, pantalla de glosario de términos, pantalla de búsqueda avanzada y pantalla de créditos. Esta multimedia brinda las siguientes funcionalidades:

- Ver imágenes de la historia de la multimedia.
- Ver video de la historia de la multimedia.
- Imprimir historia.
- Consultar la descripción de una especie.
- Ver imágenes de esa especie.

- Ampliar o restaurar imagen.
- Imprimir descripción de la especie.
- Realizar una búsqueda simple dentro de la descripción de la especie.
- Realizar una búsqueda avanzada siguiendo una serie de criterios, entre los que se encuentra: reino, especie, phylum y familia.
- Consultar el glosario de términos.
- Restaurar o maximizar la multimedia.
- Activar el sonido de la multimedia.
- Desactivar el sonido de la multimedia
- Salir de la multimedia desde cualquier pantalla.
- Ir al menú inicio.
- Mostrar los créditos una vez confirmada la salida de la aplicación.
- Saltar los créditos.

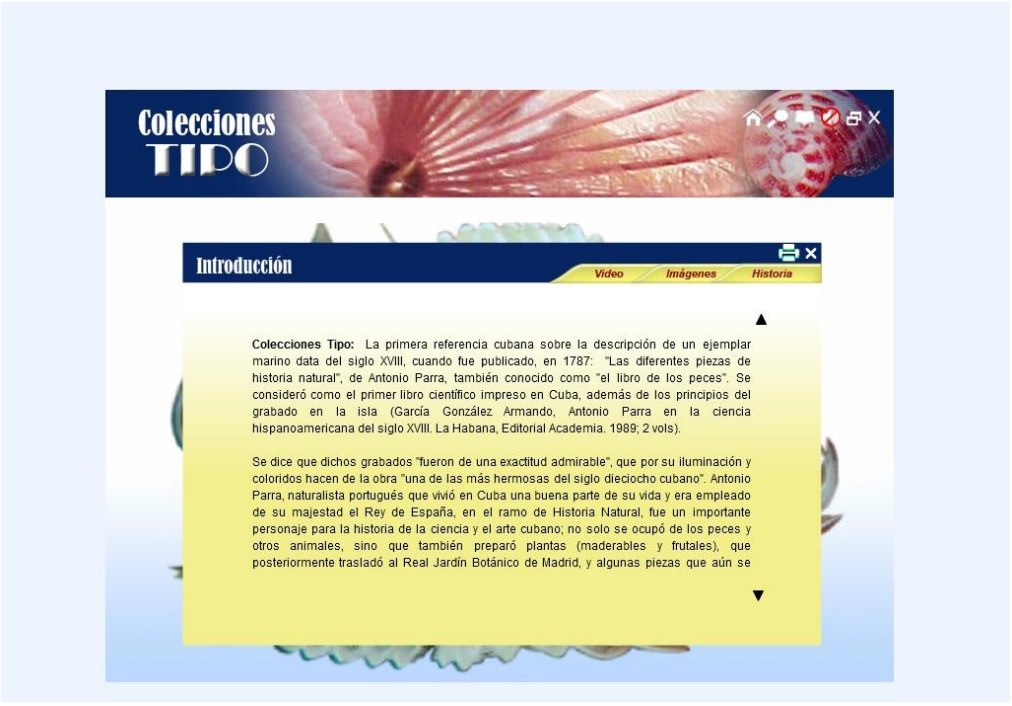


Figura 1: Pantalla de inicio de la multimedia Colecciones Tipo

Guía Náutica de Cuba

La Multimedia Guía Náutica de Cuba forma parte de los productos solicitados por la Oficina Nacional de Información Turística perteneciente al Ministerio del Turismo, con el objetivo de fortalecer la imagen turística de Cuba en la modalidad náutica. Se pretende con el producto desarrollado guiar a quienes ofrecen información al turista, así como a visitantes actuales, brindando información útil y actualizada sobre la isla. Además, se le proporciona al usuario la facilidad de entender lo que ve, encontrar lo que busca y realizar las operaciones que desee con el menor esfuerzo posible. El producto multimedia desarrollado está compuesto por cinco pantallas principales: la pantalla de inicio, pantalla de menús, pantalla de contenido, pantalla de mapa y pantalla de créditos. Esta multimedia brinda al usuario las siguientes funcionalidades:

- Mostrar video de presentación
- Mostrar breve reseña histórica sobre la náutica en Cuba.
- Navegar a través de cada una de los destinos turísticos y las modalidades.
- Mostrar el contenido referente a cada una de los destinos.
- Realizar Búsqueda de determinada información.
- Mostrar los resultados de la búsqueda.
- Desactivar y activar audio de fondo.
- Imprimir contenido incluido en la aplicación.
- Representar en Mapa las bases náuticas, zonas de buceo y marinas con las que cuenta la Isla.
- Abandonar la aplicación desde cualquier pantalla.
- Mostrar créditos una vez confirmada la salida de la aplicación.
- Mostrar imágenes de cada uno de los destinos turísticos que componen la multimedia.
- Permitir restaurar o maximizar la aplicación.
- Mostrar galería (Fotos y videos).

Figura 2: Pantalla de inicio de la multimedia Guía Náutica de Cuba

La Caja Mágica

La Caja Mágica constituye una colección de *software* educativo multiplataforma, creada para dotar al Sistema Educacional Venezolano de 14 *software* educativos, promoviendo así la formación integral de los estudiantes de la enseñanza primaria. Su desarrollo estuvo a cargo de la Universidad de las Ciencias Informáticas, en colaboración con el Ministerio de Educación de Cuba. Cada producto de la colección está dividido en seis módulos fundamentales: Temas, Ejercicios, Juegos, Mediateca, Resultados y Profesor, además del módulo General. Estos productos abarcan contenidos de las asignaturas de Matemática, Lengua Española, Ciencias Naturales e Informática. Las tecnologías y herramientas utilizadas para su implementación son de código abierto. Esta colección brinda las funcionalidades siguientes:

- Seleccionar tema por contenido.
- Visualizar imágenes.
- Reproducir videos.
- Seleccionar tipo de ejercicio por tema.

- Seleccionar cantidad de ejercicios a realizar.
- Consultar artículos de interés y glosario de términos en el módulo Mediateca.
- Consultar resultados.
- Activar o desactivar sonido.
- Salir de la aplicación desde cualquier pantalla.
- Abrir o cerrar mascota.
- Imprimir contenido.
- Realizar búsquedas.
- Recibir ayuda.
- Visualizar créditos de los autores de la aplicación.

Figura 3: Pantalla de inicio del módulo Ciencias Naturales de la Caja Mágica

Las multimedia Colecciones Tipo y Guía Náutica de Cuba fueron desarrolladas con el objetivo de ser utilizadas dentro de nuestro país, lo que no supone pago de licencia para su uso. Sin embargo, su

implementación se sustenta en tecnologías privativas. Se utilizaron las herramientas que provee el paquete Adobe CS6, tomando como entorno de creación a Adobe Flash y como reproductor o máquina virtual a Adobe Flash Player. Esta tecnología, a pesar de ser muy utilizada en el proceso de desarrollo de multimedia, resulta muy costosa para el país, aspecto que tiene a favor La Caja Mágica, que está basada en tecnologías libres. El análisis realizado permitió determinar las funcionalidades genéricas en las multimedia, que serán utilizadas como base para el desarrollo de los componentes básicos que conformarán el marco de trabajo.

Las funcionalidades definidas son:

- Realizar búsquedas.
- Consultar glosario de términos.
- Seleccionar menú.
- Visualizar y ampliar imagen.
- Reproducir videos.
- Gestionar mapas.

Los componentes básicos son:

- Visor de imágenes.
- Visor de videos.
- Buscador simple y avanzado.
- Menú.
- Glosario de términos.
- Trabajo con mapas.

1.4 Tecnologías y herramientas

Para el proceso de 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 el documento Arquitectura de Software de la versión 1.0 del marco de trabajo de desarrollo de multimedia con tecnologías libres.

1.4.1 Metodología de desarrollo de software

Una metodología de software propone un conjunto de directrices por las que se debe regir todo equipo de desarrollo de software, para obtener un producto con calidad y acorde a los requerimientos y necesidades del cliente. Existen numerosos ejemplos de metodologías de desarrollo de software, tanto ágiles como tradicionales. En la línea de multimedia se emplea la metodología de desarrollo Proceso Unificado Ágil (AUP), la cual constituye la guía propuesta por la UCI para ser aplicada en todos los proyectos productivos, con el fin de lograr una uniformidad en el trabajo que se realiza en cada uno de ellos.

Proceso Unificado Ágil (AUP)

El tiempo constituye un factor determinante en cualquier proyecto de desarrollo de software. Por ello, la correcta planificación del mismo, garantiza la obtención de un producto con altos niveles de calidad. Las estimaciones del tiempo unidas a la planeación del proyecto, representan aspectos en los que divergen todas las metodologías de desarrollo de software (Rodríguez, 2014). En busca de la homogeneidad en los proyectos productivos de la Universidad, se decidió escoger una metodología que converja en las particularidades de cada uno, sin alejarse mucho de lo que hasta el momento se estaba proponiendo en la Universidad. De esta manera surge el Proceso Unificado Ágil o Agile Unified Process (AUP), que constituye una versión reducida de Proceso Unificado del Rational (RUP). AUP es una metodología que aplica técnicas ágiles, entre las que se incluyen (Rodríguez, 2014):

Técnicas de desarrollo que propone AUP

- El desarrollo dirigido por pruebas.
- La modelación ágil.
- Gestión de cambios ágil.
- Refactorización de base de datos para mejorar la productividad.

Al igual que RUP, AUP cuenta con cuatro fases que se desarrollan de manera consecutiva (Rodríguez, 2014):

Fases del ciclo de vida de AUP

1. Inicio.
2. Elaboración.
3. Construcción.
4. Transición.

Durante el proceso de adaptación de esta metodología a los proyectos desarrollados en la universidad, se realizó una modificación a las fases que propone AUP, quedando de la siguiente manera: se mantiene la **fase de inicio**. La elaboración, construcción y transición, se agrupan en la **fase ejecución** y se incorpora la **fase de cierre**.

AUP define siete disciplinas. De ellas cuatro son dirigidas a la ingeniería y tres a la gestión de proyectos. La variación que realiza la Universidad define nuevas disciplinas para AUP, las cuales se muestran a continuación (Rodríguez, 2014):

Disciplinas de AUP

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

Las últimas tres disciplinas que propone AUP: gestión de configuración, gestión de proyectos y entorno, quedan cubiertas por la gestión de la configuración, la planeación del proyecto y el monitoreo y control de proyecto del modelo CMMI-DEV[1] en su versión 1.3.

AUP propone nueve roles para el desarrollo de un proyecto. La adaptación que le realiza la Universidad a la metodología propone once roles, manteniendo algunos de AUP y unificando otros para una mejor organización del equipo de desarrollo. A continuación se muestran los **roles** resultantes (Rodríguez, 2014):

- Jefe de proyecto.
- Planificador.
- Analista.
- Arquitecto de la información.
- Desarrollador.

[1] Del inglés Capability Maturity Model Integration.

- Administrador de la configuración.
- Stakeholder (cliente/proveedor de requisitos).
- Administrador de calidad.
- Probador.
- Arquitecto de software.
- Administrador de base de datos.

1.4.2 Lenguajes de programación

Un lenguaje de programación es una herramienta que permite crear programas y *software*. Existen dos grandes grupos de lenguajes de programación: los del lado del cliente y los del lado del servidor. Para el desarrollo de la presente investigación sólo se utilizarán **lenguajes del lado del cliente**, específicamente los que se describen a continuación:

HTML: *Según la W3C[2], HTML[3] es "Un lenguaje comúnmente utilizado para la publicación de hipertexto en la Web y desarrollado con la idea de que cualquier persona o tipo de dispositivo pueda acceder a la información en la Web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento."* (W3C, 2015).

La versión 5 de HTML constituye el resultado de la unión de CSS3, HTML y JavaScript, donde HTML se encarga de la estructura. Luego CSS[4] lleva esa estructura a la pantalla de manera tal que resulte atractiva a la vista y JavaScript se encarga de proveer dinamismo y una mayor interacción de la aplicación con el usuario (Gauchat, 2012). Las numerosas ventajas que ofrece HTML5 con respecto a su versión anterior, lo ubican en el presente y futuro del desarrollo web, entre ellas se encuentran:

- Incorporación de etiquetas como, <meta> para la definición del tipo de caracteres, <header> para brindar información introductoria, <nav> para la navegación, <section> para una sección del documento y <footer> para representar el fin del cuerpo del documento o puede ser usado también dentro del cuerpo, para darle final a una sección.
- Incluye etiquetas para incorporar contenido multimedia como <*audio*> para audio y <*video*> para video, las cuales permiten la reproducción por parte del navegador de este tipo de contenido.

[2] Del inglés World Wide Web Consortium.
[3] Del inglés HyperText Markup Language.
[4] Del inglés Cascading Stylesheets.

- El doctype sirve para indicar al navegador a qué reglas de escritura obedece el código fuente de la página HTML. HTML5 lo declara con una sencilla sintaxis: <!DOCTYPE html>, lo que posibilita una mejor memorización (Lancker, 2013).
- Incorpora nuevos atributos, que facilitan el trabajo con HTML, por ejemplo: **ping,** que será particularmente útil para elaborar las estadísticas de un sitio web; **target**, especifica al navegador la forma de visualizar el contenido del enlace. Estas pueden ser en una nueva pestaña, (target="_blanck") o en la misma ventana de la página de inicio del enlace, (target="_self") (Lancker, 2013).

JavaScript: *"Es el lenguaje interpretado más utilizado, principalmente en la construcción de páginas Web, con una sintaxis muy semejante a Java y a C. Pero, al contrario de Java, no se trata de un lenguaje orientado a objetos propiamente dicho, sino que éste está basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad..." (Flanagan, 2007).*
JavaScript es un lenguaje ampliamente difundido y utilizado. Esta condición se debe, en gran medida, a las opciones que brinda, entre las que se incluyen:

- La máxima interactividad entre el usuario y la página.
- La verificación de los datos introducidos por el usuario, antes de enviar el formulario al servidor.
- La ejecución de pequeñas cantidades de información al igual que en una base de datos.
- El manejo de *applets* y *plugins* dentro de múltiples marcos de HTML.
- El pre procesado de información antes de enviarla al servidor.

CSS: *"…es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo y bordes."* (Gauchat, 2012). Con su utilización se logra un ahorro de código considerable, ya que si los estilos se repiten no es necesario repetir todo el código, además que los estilos que ya han sido creados pueden ser utilizados en todo el sitio web. En contrapartida con las características antes mencionadas, se encuentra que CSS no es soportado por todos los navegadores, además de que estos últimos difieren en el estándar a utilizar, lo que trae consigo resultados diferentes en cada navegador.

"…*CSS3 no solo cubre diseño y estilos web sino también forma y movimiento*." (Gauchat, 2012) Esta nueva versión incorpora novedosas propiedades que permiten crear efectos visuales y dinámicos vitales en la web

actual, razón por la cual será utilizada en el desarrollo de la propuesta de solución. Dentro de las principales ventajas que presenta esta versión con respecto a la anterior se encuentran (Gauchat, 2012):

- Logra las esquinas redondeadas con la propiedad **border-radius.**
- Crea sombras en los elementos (**box-shadow).**
- Incorpora la posibilidad de darle sombra a los textos **(text-shadow).**
- Brinda la posibilidad de inserción de diferentes tipos de fuente con **@font-face.**
- Crea fondos con degradados y con imágenes múltiples.
- Aplica transformaciones, transiciones y animaciones a los elementos.
- Crea sitios web que se adapten a distintos tamaños de pantalla.

1.4.3 Framework de JavaScript

"El término marco de trabajo, hace referencia a una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. También se puede ver como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta" (Valbuena, 2014).

Los estudios realizados en la presente investigación muestran la existencia de disímiles marcos de trabajo para el lenguaje JavaScript. Dentro de este amplio grupo, se destacan AngularJS y JQuery, librerías de código abierto que facilitan el desarrollo de aplicaciones web, utilizando el lenguaje JavaScript.

AngularJS: "*Es el framework de JavaScript de código abierto que utiliza el Modelo-Vista-Controlador (MVC), el enlace de datos, plantillas del lado del cliente y la inyección de dependencias para crear una estructura muy necesaria para la construcción de aplicaciones web"* (Green, Seshadri, 2013). AngularJS se caracteriza por no ocultar el HTML y el CSS, toma sus fortalezas y las extiende volviéndolas adecuadas para la descripción de vistas dinámicas. Además, utiliza un grupo de directrices para la creación de nuevas etiquetas HTML o nuevos atributos (Valbuena, 2014).

JQuery: es una biblioteca de JavaScript rápida, pequeña y rica en funciones (CHAFFER, SWEDBERG, 2010). *"Es un framework[5] JavaScript libre y Open Source, del lado del cliente, que se centra en la interacción*

5 El autor de esta fuente denomina a JQuery como un framework, sin embargo en la investigación es denominado como librería o biblioteca, que es la denominación dada por los desarrolladores en la página oficial http://jquery.com/.

entre el DOM[6]*, JavaScript, AJAX*[7]*y HTML*[8]*."* (Lancker, 2012). Tiene como objetivo simplificar los comandos comunes de JavaScript. De hecho, el lema de JQuery es <<Escribir menos para hacer más>> (Lancker, 2012). Dentro de sus principales características se encuentran:

- Permite manipular fácilmente el DOM de una página web.
- Permite manejar de una manera sencilla los eventos de los elementos que forman una página web.
- Permite modificar, eliminar o acceder a los estilos visuales de una página web.
- Permite el uso de AJAX.
- Permite manipular información en formato JSON[9].
- Puede ser extendido a través del uso de *plugin*.
- Posee compatibilidad con navegadores como Mozilla Firefox, Internet Explorer, Safari, Opera y Google Chrome.

La utilización de estas librerías en el desarrollo de la propuesta de solución estuvo respaldada por las ventajas que brindan a la programación con JavaScript, ventajas que posibilitan que sean preferidas en la web.

1.4.4 Framework CSS

En la actualidad, el uso de marcos de trabajo que faciliten el diseño web se ha hecho muy común entre los desarrolladores. Un ejemplo de estos lo constituye **Twitter Bootstrap**[10], el cual provee una colección de herramientas de software libre para la creación de sitios y aplicaciones web. Contiene plantillas de diseño basadas en HTML y CSS con tipografías, formularios, botones, gráficos, barras de navegación y demás componentes de interfaz, así como extensiones opcionales de JavaScript (Bootstrap, 2015).

La versión que se utilizará en la propuesta de solución será la 3.3.4 que fue lanzada el 16 de marzo del 2015. Cada nueva versión de Bootstrap se encarga de corregir errores o mejorar funcionalidades de los

6 Del inglés Document Object Model.
7 Del inglés Asynchronous JavaScript And XML. Proporciona a los desarrolladores la capacidad de crear interfaces de usuario más sofisticadas y con respuesta casi inmediata (Mellado, 2008).
8 Del inglés HyperText Markup Language.
9 Del inglés JavaScript Object Notation, Página oficial del estándar: http://www.json.org/.
10 Página oficial http://getbootstrap.com/

módulos de la versión que la antecede. Por ejemplo Twitter Bootstrap v3.3.4 ofrece correcciones para algunos errores significativos en el plugin Modal y para un algunos bugs molestos en el plugin ScrollSpy.

1.4.5 Herramienta CASE [11]

Visual Paradigm: es una herramienta CASE, que soporta el modelado por UML[12] y el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue (Visual Paradigm, 2015).

El uso de esta herramienta proporciona disímiles facilidades, entre las que se hayan (Visual Paradigm, 2015):

- Generación de código y la base de datos a partir de los diagramas UML realizados.
- Permite realizar el proceso de Ingeniería Inversa.
- Posibilita la reproducción automática de informes en formato como PDF, Word o HTML, proporcionando la estandarización de la documentación.

La versión a utilizar será la 8.0 que es la más actualizada en los servidores de la UCI. Esta versión facilita el modelado de UML, proporcionando herramientas específicas para ello y estandarizando la documentación. Además controla que el modelado con UML sea correcto y posibilita visualizar un mismo elemento en varios diagramas, evitando duplicidades. La integración con otras aplicaciones ofimáticas y la reutilización de modelos, constituyen otras de las ventajas que ofrece Visual Paradigm 8.0 (Visual Paradigm, 2015).

1.4.6 Entorno integrado de desarrollo

NetBeans: es un entorno de desarrollo integrado (IDE), una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. Es un producto de código abierto, libre y gratuito sin restricciones de uso (Oracle Corporation**,** 2015).

[11] CASE por sus siglas del inglés Computer Aided Software Engineering.
[12] Del inglés Unified Modeled Language

El NetBeans IDE posibilita el rápido y fácil desarrollo de aplicaciones Java de escritorio, móviles y aplicaciones web, así como HTML5 con HTML, JavaScript y CSS. Además, provee un gran conjunto de herramientas para desarrolladores PHP, C y C++ (Oracle Corporation**,** 2015).

La versión 8 es la que será utilizada en el desarrollo de la propuesta de solución. Esta, además de incorporar un grupo de novedosas herramientas para HTML5, en particular para AngularJS, provee facilidades para el completamiento de código (Oracle Corporation, 2015). La selección fue respaldada por la familiarización y experiencia de trabajo con esta herramienta.

1.5 Conclusiones del capítulo

El estudio realizado, permitió elaborar el marco teórico que sustenta la investigación. El análisis de las soluciones similares existentes, posibilitó determinar las funcionalidades genéricas en las multimedia, que serán utilizadas como base para el desarrollo de los componentes básicos que conformarán el marco de trabajo. Además, se describió la metodología de desarrollo de *software* AUP, los lenguajes de programación HTML5, CCS3 y JavaScript, así como la herramienta de modelado Visual Paradigm y el IDE NetBeans, que constituyen las herramientas y tecnologías que serán empleadas en el desarrollo de la propuesta de solución.

Capítulo 2: Descripción de la propuesta de solución

2.1 Introducción

Luego de realizar la fundamentación teórica que sustenta la presente investigación, quedaron sentadas las bases para describir la propuesta de solución. En este capítulo se muestra el modelo conceptual o de dominio, así como una explicación de los conceptos relacionados en el mismo. Se describe la propuesta de solución y se realiza el levantamiento de los requisitos funcionales y no funcionales. Se caracteriza la arquitectura y los patrones de diseño empleados en el desarrollo de la propuesta de solución. Además, se muestra el modelo de clases del diseño.

2.2 Modelo de dominio

Un modelo de dominio o modelo conceptual muestra clases conceptuales significativas en un dominio (Larman, 2003). Es un artefacto ingenieril que brinda un mejor entendimiento de los conceptos que se manejan dentro de un sistema informático.

A continuación se describen cada uno de los conceptos que se relacionan en el modelo de dominio.

- **Marco de trabajo:** es una estructura de *software* compuesta por componentes personalizables y reutilizables, que permitirán desarrollar multimedia con tecnologías libres.
- **Multimedia:** sistema informático interactivo, que combina coherentemente, imágenes, textos, sonidos y animaciones para expresar un conocimiento de manera dinámica al usuario.

- **Componentes básicos:** término que define los componentes que resultan comunes en las multimedia (visor de imágenes y videos, buscador, glosario de términos, menú, trabajo con mapas).
    - Visor de imágenes: componente que muestra las imágenes de cinco maneras diferentes. Además, permite ampliarlas para obtener una mejor vista de estas.
    - Visor de videos: permite reproducir, detener o pasar un video.
    - Buscador: brinda la posibilidad de realizar búsquedas siguiendo un criterio determinado.
    - Glosario de términos: componente que ayuda a entender los términos que son manejados dentro de una multimedia.
    - Menú: contribuye con la navegabilidad dentro de una multimedia, brindando diferentes maneras de acceder a los restantes componentes.
    - Trabajo con mapas: componente que permite conocer la localización de un lugar ubicado por el usuario en el mapa. Además posibilita ampliar o disminuir el mapa.
- **Medios de información:** constituyen los elementos que integran una multimedia. Dentro de estos medios se destacan, las imágenes, las animaciones, los textos y el sonido.
    - Imagen: medio que permite transmitir una información en la multimedia. Puede ser en formato jpg, png o gif.
    - Video: medio que muestra una secuencia animada de imágenes. Puede ser en formato mp4 o avi.
    - Texto: medio que está presente en toda la multimedia, en la descripción de las imágenes y los videos, en el buscador, en el glosario de términos y en el menú.
    - Iconográfico: medio que se utiliza en el componente mapa para diferenciar los puntos que serán ubicados en el mismo.

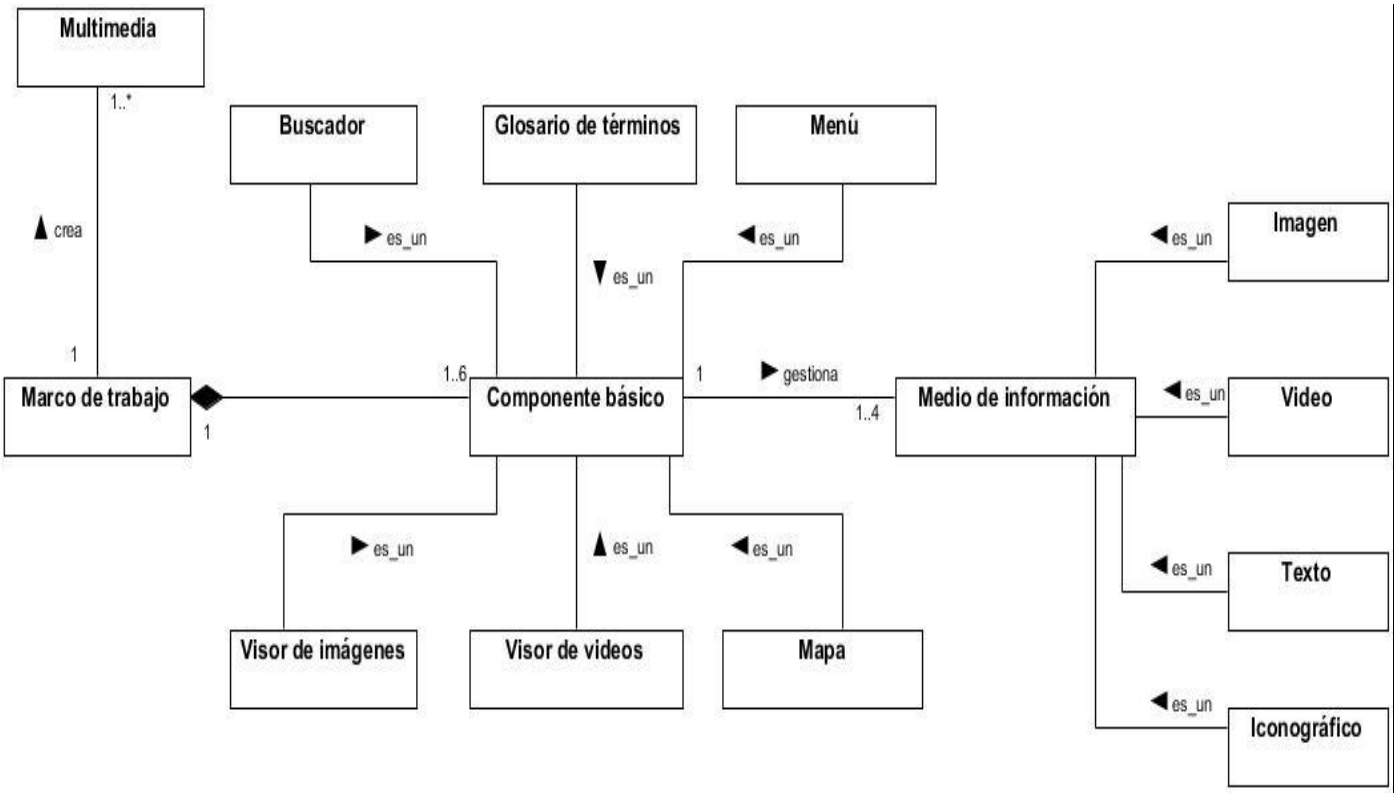


Figura 4: Diagrama de dominio de la propuesta de solución

2.2 Descripción de la propuesta de solución

La propuesta de solución está dirigida a desarrollar los componentes básicos (visor de imágenes, visor de video, buscador, glosario de términos, menú y mapas) que posteriormente serán incorporados a un marco de trabajo que será libre y multiplataforma, con el que podrá interactuar cualquier usuario.

El desarrollo de estos componentes básicos contribuirá en gran medida con el proceso de creación de multimedia, incorporándole numerosos beneficios. Entre las principales ventajas que ofrece su utilización, se encuentra la posibilidad de ser ejecutados en cualquier sistema operativo y en diferentes dispositivos. Además, pueden ser reutilizables, característica que trae consigo una reducción de costos y un aumento en la calidad del producto obtenido.

2.3 Requisitos del sistema.

Se entiende por requisito de *software* a la capacidad que debe alcanzar o poseer un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal (Sánchez, Sicilia y García, 2012).

Existen dos tipos de requisitos: los funcionales y los no funcionales. A continuación se describe cada uno y se muestran los definidos para la propuesta de solución.

2.3.1 Requisitos funcionales.

Un requisito funcional especifica una función que un sistema o componente de un sistema debe ser capaz de llevar a cabo (Sánchez, Sicilia y García, 2012).

La propuesta de solución debe satisfacer una serie de requisitos funcionales que se muestran a continuación:

RF1. Listar temas de galerías de imágenes.

RF2. Seleccionar tema de galería de imágenes.

RF3. Adicionar descripción a la imagen.

RF4. Ampliar imagen.

RF5. Listar temas de galerías de videos.

RF6. Seleccionar tema de galería de videos.

RF7. Adicionar descripción al video.

RF8. Listar temas de menús.

RF9. Seleccionar tema de menú.

RF10. Realizar búsqueda de información.

RF11. Mostrar glosario de términos.

RF12. Cargar datos del json.

RF13. Guardar datos en el json.

RF14. Gestionar mapa.

A continuación se muestra la descripción de los requisitos del componente visor de videos. Las restantes descripciones, se pueden consultar en los anexos.

Tabla 1: Descripción del requisito Listar temas de galería de videos.

Precondiciones	Deben existir videos para visualizar en la galería.	
Flujo de eventos		
Flujo básico <<Listar temas de galerías de videos>>		
1.	El desarrollador selecciona el componente visor de videos.	
2.	Se muestran los cuatro temas disponibles para el visor de videos.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran los cuatro temas predefinidos para el componente visor de videos.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1		
2		
Pos-condiciones		
1		
Validaciones		
1	Se validan los videos subidos	
Conceptos	**Galería de videos**	Componente que permite visualizar videos.
	Video	Título: Palabra o conjunto de palabras que dan información sobre el video. Descripción: Breve comentario sobre el video. Formato: Forma del fideo “webm, ogg y mp4”. Tamaño: Formato del video. Dirección: Lugar donde se encuentra el video.
Requisitos especiales	Se debe tener en cuenta el tamaño de los videos y el formato de los mismos	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable.	

Tabla 2: Descripción del requisito Seleccionar tema de galería de videos.

Precondiciones	Deben existir videos para visualizar en la galería. Debe haberse ejecutado el requisito funcional: Listar temas de galerías de videos.
Flujo de eventos	
Flujo básico <<Seleccionar tema de galería de videos>>	
1.	El desarrollador selecciona el tema de galería de videos que desee.

2.	Se muestra el tema seleccionado.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran los videos según el tema seleccionado.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan los videos subidos	
Conceptos	**Galería de videos**	Componente que permite visualizar videos.
	Video	Título: Palabra o conjunto de palabras que dan información sobre el video. Descripción: Breve comentario sobre el video. Formato: Forma del fideo “webm, ogg y mp4”. Tamaño: Formato del video. Dirección: Lugar donde se encuentra el video.
Requisitos especiales	Se debe tener en cuenta el tamaño de los videos y el formato de los mismos	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable.	

Tabla 3: Descripción del requisito Adicionar descripción al video.

Precondiciones	Deben existir imágenes para visualizar en la galería. Debe haberse ejecutado el requisito funcional: Listar temas de galerías de videos Debe haberse ejecutado el requisito funcional: Seleccionar tema de galería de videos
Flujo de eventos	
Flujo básico <<Adicionar descripción al video>>	
1.	El desarrollador selecciona el archivo json correspondiente al componente video y modifica el atributo descripción.
2.	Se muestra la descripción del video.
Pos-condiciones	
1.	Se ejecuta la funcionalidad y se muestra la descripción del video.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>	

1.		
2.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan los videos subidos y la descripción	
Conceptos	**Galería de videos**	Componente que permite visualizar videos.
	Video	Título: Palabra o conjunto de palabras que dan información sobre el video. Descripción: Breve comentario sobre el video. Formato: Forma del fideo “webm, ogg y mp4”. Tamaño: Formato del video. Dirección: Lugar donde se encuentra el video.
Requisitos especiales	Se debe tener en cuenta el tamaño del video y el formato del mismo.	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable. Poder insertar una descripción a un video sin tener que acceder al archivo json.	

2.3.2 Requisitos no funcionales.

Los requisitos no funcionales en un sistema son aquellos que especifican aspectos técnicos que debe incluir la aplicación (Sánchez, Sicilia y García, 2012).

- **Usabilidad**

RNF1. Navegadores para acceder: Internet Explorer 7.0 o superior, Mozilla Firefox 3.6 o superior, Google Chrome y Node web kit.

- **Soporte**

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

- **Diseño e implementación**

RNF3. Los lenguajes de programación que se utilizarán son: JavaScript, HTML5, CCS3. El framework que se empleará para el desarrollo será: AngularJS y para las vistas se usará JQuery y Twitter Bootstrap v3.3.4.
RNF4. Se utilizará el patrón arquitectónico MVC que establece el framework de desarrollo Angular.

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

RNF5. Cada componente debe estar acompañado de un documento que refleje la descripción de sus funcionalidades, cómo usarlas y extenderlas.

RNF6. Cada componente debe estar acompañado de un documento que refleje el *software* necesario para instalar la aplicación, así como los pasos para su puesta en funcionamiento.

- **Requisitos de licencia**

RNF7. 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**

RNF8. Los componentes deber ser independientes de plataforma. Deben ejecutarse tanto en Microsoft Windows como en GNU/Linux y Mac OS.

- **Reusabilidad**

RNF9. Los componentes que conforman el marco de trabajo deben poder ser reutilizados en otras aplicaciones, brindando para ello una interfaz de comunicación.

- **Escalabilidad**

RNF10. Cada componente debe responder correctamente ante la ampliación del diseño arquitectónico, de datos o procedimental sin afectar más de dos clases.

2.4 Descripción de la arquitectura.

El término arquitectura en Informática, se refiere a la manera de estructurar una computadora, sistema operativo o software. Una arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes, que brindan el marco de referencia necesario para servir de guía a la construcción de un software para un sistema de información (Alegsa, 2015).

La arquitectura de la propuesta de solución está determinada por el uso del marco de trabajo AngularJS, que permite el desarrollo de aplicaciones web siguiendo una arquitectura Modelo Vista Controlador (MVC).

El MVC es un patrón arquitectónico, cuya idea central es la clara separación del código entre: la gestión de los datos (modelo), la lógica de la aplicación (controlador) y la presentación de los datos al usuario (vista) (Green, Seshadri, 2013). En AngularJS la vista es el modelo de objetos de documento (DOM), los controladores son las clases JavaScript y los datos del modelo son almacenados en las propiedades de los objetos (Green, Seshadri, 2013).

El funcionamiento del MVC se basa principalmente en que las vistas obtienen los datos del modelo para mostrarlos al usuario. Cuando un usuario interactúa con la aplicación, ya sea dando clic o escribiendo en la

misma, el controlador responde a estas acciones, cambiando los datos en el modelo. Finalmente, el modelo notifica a la vista los cambios ocurridos, de modo que esta pueda actualizar lo que muestra (Green, Seshadri, 2013).

2.5 Patrones de diseño.

"Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas." (Larman, 2003, p.204) Los patrones no se proponen descubrir ni expresar nuevos principios de la Ingeniería del *Software,* al contrario, intentan codificar el conocimiento, las expresiones y los principios ya existentes (Larman, 2003). Se definen tres grupos de patrones: de creación, de comportamiento y los estructurales (Gamma, Helm, Johnson, y otros, 1994), los cuales son conocidos en el mundo del *software* como patrones GoF[13]. Además existen los patrones generales para asignar responsabilidades (GRASP[14]), que constituyen cuestiones muy básicas, comunes en el diseño (Larman, 2003). A continuación se describen los que fueron utilizados en la propuesta de solución.

2.5.1 Patrones GRASP

- **Bajo acoplamiento:** constituye un principio a tener en cuenta en todas las decisiones de diseño. "*El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos"* (Larman, 2003, p.215). Un elemento con bajo o débil acoplamiento no depende de demasiados otros elementos (Larman, 2003). El bajo acoplamiento se evidencia en que las clases controladoras de cada componente heredan de una única clase controladora general. Además, las vistas están separadas del modelo, lo que propicia la baja dependencia entre clases.
- **Alta cohesión:** patrón que define que un elemento posea pocas responsabilidades y que estas estén altamente relacionadas (Larman, 2003). *La cohesión es una medida de la fuerza con la que se relacionan y del grado de focalización de las responsabilidades de un elemento* (Larman, 2003, p.217). La alta cohesión se evidencia en las clases controladoras, que cuentan con una serie de funcionalidades relacionadas entre sí.
- **Experto:** este patrón resuelve el problema de "*asignar una responsabilidad al experto en información- la clase que tiene la información necesaria para realizar la responsabilidad-"*(Larman, 2003, p.207). El

[13] Del inglés The Gand of Four
[14] GRASP: del inglés General Responsibility Assignment Software Patterns

uso de este patrón se evidencia en la clase controladora general, que provee el servicio leer_json a las restantes clases controladoras.

- **Controlador:** este patrón resuelve el problema de asignar la responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase. Un controlador sirve como intermediario entre una interfaz y la acción que se desee ejecutar (Larman, 2003). Este patrón se evidencia en las clases que conforman la capa controlador del MVC. Ejemplos de estas clases lo constituyen fwCGeneral, fwCVisorDeImagenes, y fwCVisorDeVideos.

2.6 Modelo de clases del diseño.

El modelo de diseño es un artefacto ingenieril que incluye diagramas de interacción, de paquetes y de clases (Larman, 2003). Los diagramas de clases del diseño representan las especificaciones e interfaces de *software* (Larman, 2003). En la propuesta de solución se realizaron cinco diagramas de clases del diseño. A continuación se muestran tres de ellos, los restantes se incluyen en los anexos.

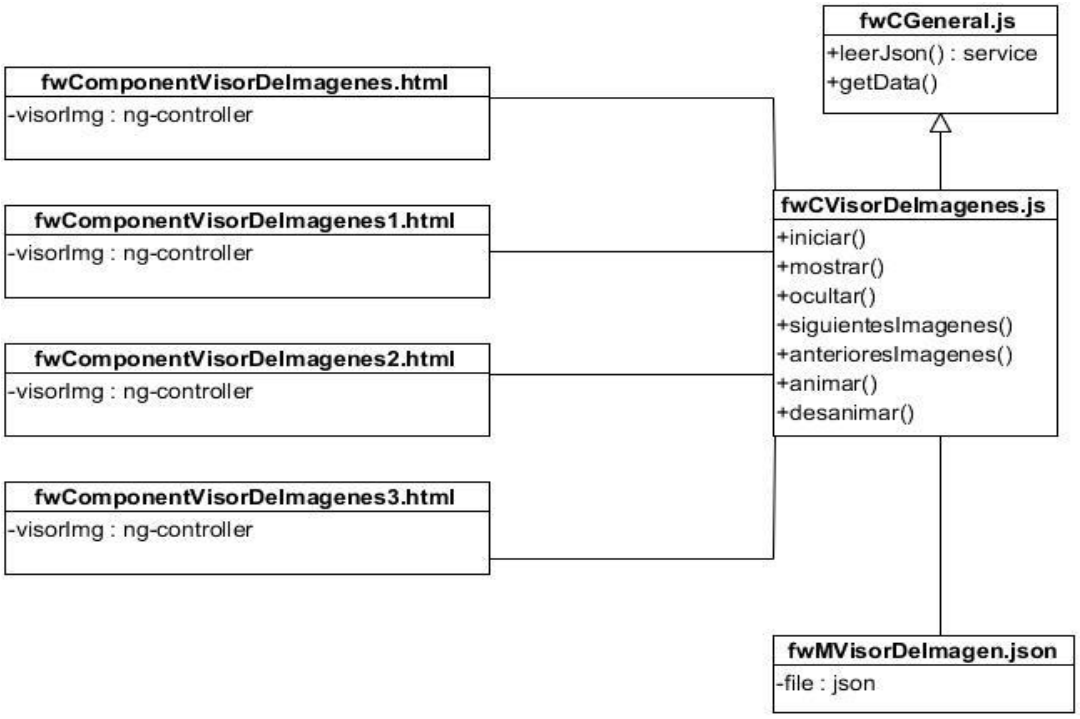


Figura 5: Diagrama de clases del diseño. Componente Visor de imágenes

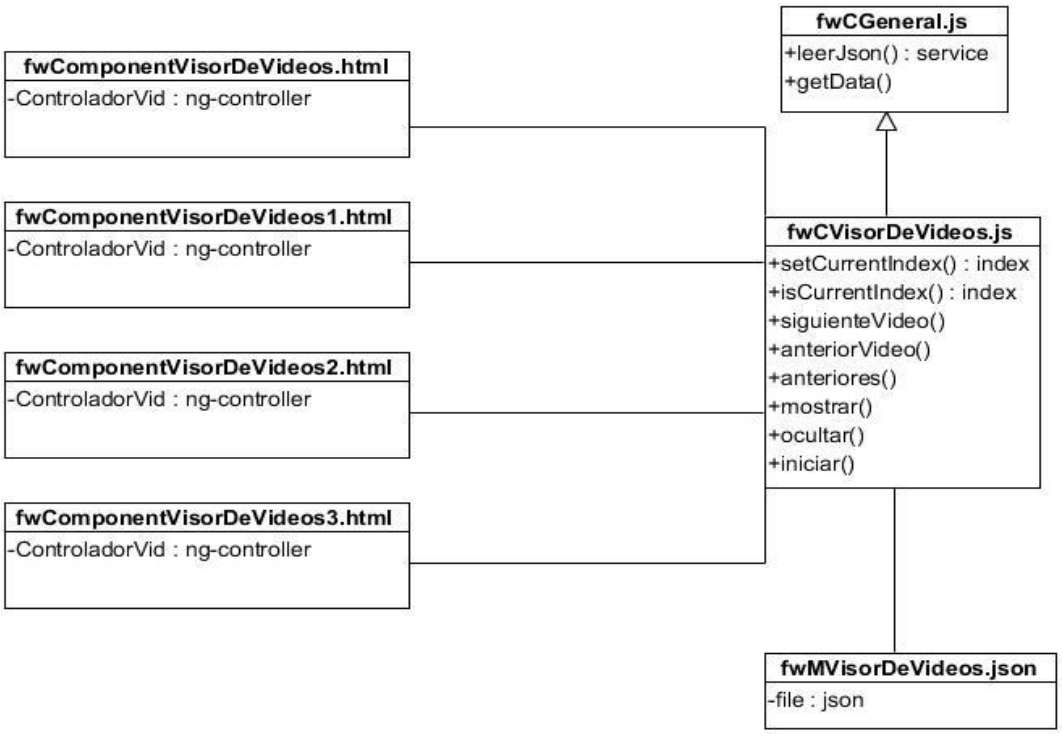


Figura 6: Diagrama de clases del diseño. Componente Visor de videos

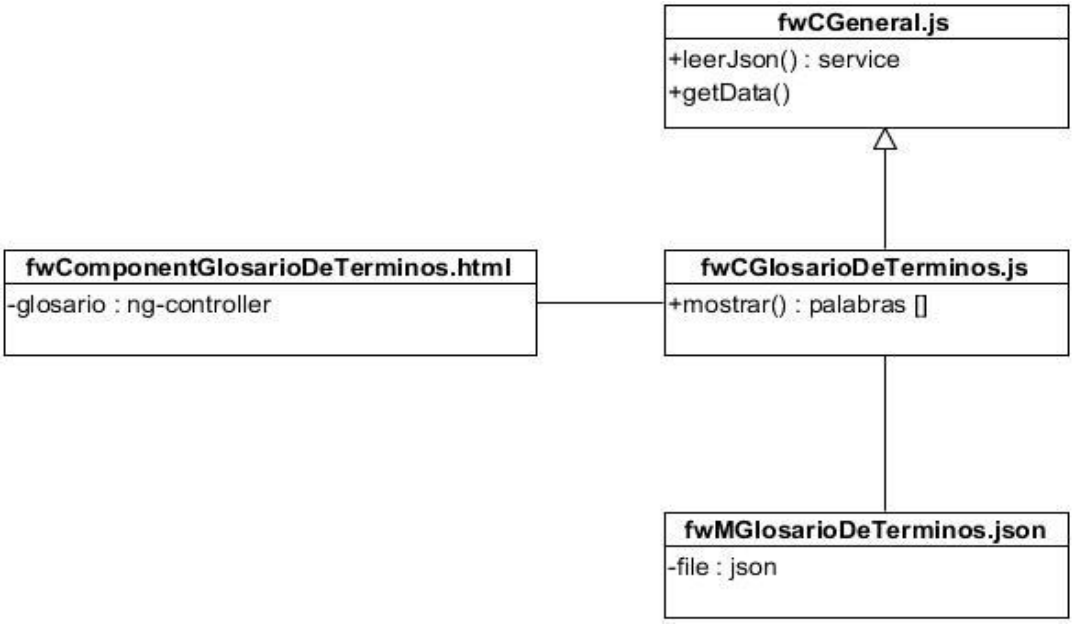


Figura 7: Diagrama de clases del diseño. Componente Glosario de términos

2.7 Conclusiones del capítulo

Los artefactos ingenieriles generados permitieron describir la propuesta de solución, que posteriormente servirá de guía para la implementación. La arquitectura MVC y los patrones de asignación de responsabilidades (GRASP) le proporcionan al equipo de desarrollo un conjunto de buenas prácticas que contribuyen con la obtención de los resultados esperados. La confección de los diagramas de clases del diseño contribuyó a entender mejor la relación existente entre las clases que conforman la propuesta de solución.

Capítulo 3: Implementación y pruebas.

3.1 Introducción

Luego de haber realizado el diseño de la propuesta de solución, como parte del flujo de trabajo de modelado de la metodología de desarrollo AUP, se puede proseguir a desarrollar la implementación de la misma. En este capítulo se muestra el diagrama de componentes de la propuesta de solución, que brindará un mejor entendimiento de la misma. Además, se realiza una breve descripción de los estándares de codificación que serán utilizados para lograr una mayor organización en el código de la propuesta de solución. Se muestran las interfaces de usuario y se describen las pruebas que serán aplicadas a la propuesta de solución para su validación, brindando los resultados obtenidos.

3.2 Implementación

La implementación se empieza con los resultados del diseño y se implementa el sistema en términos de componentes (Booch, Jacobson, Rumbaugh, 2000). En esta fase se generan una serie de artefactos que componen el modelo de implementación, entre los que se encuentra el diagrama de componentes.

3.2.1 Diagrama de componentes

El diagrama de componentes muestra cómo el sistema está divido en componentes y la relación existente entre ellos. Los componentes pueden ser: archivos, bibliotecas, tablas o documentos, donde cada uno es representado por un estereotipo estándar. A continuación se muestran los estereotipos predominantes dentro de un diagrama de componentes (Booch, Jacobson, Rumbaugh, 2000):

Tabla 4: Estereotipos comúnmente utilizados en un diagrama de componentes

Estereotipo	Descripción
<<executable>>	Programa que puede ser ejecutado en un nodo.
<<file>>	Fichero que contiene código fuente o dato.
<<library>>	Librería estática o dinámica.
<<table>>	Tabla de la base de datos.
<<document>>	Documento.

A continuación se muestra el diagrama de componentes de la propuesta de solución:

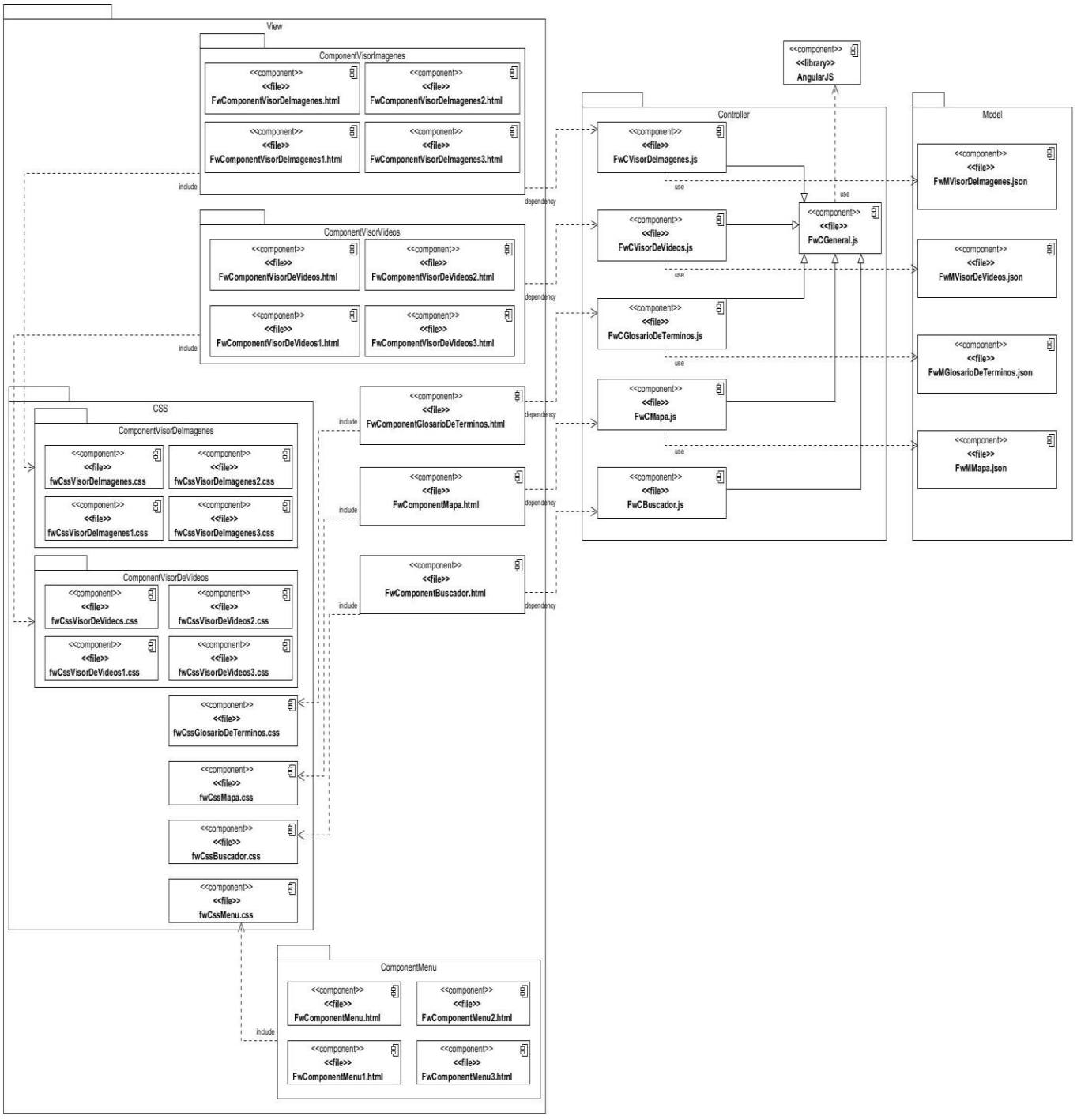


Figura 8: Diagrama de componentes de la propuesta de solución

3.2.2 Estructura de la propuesta de solución

El MVC posee dentro de sus ventajas la división del código fuente en tres capas fundamentales: la vista, el controlador y el modelo. Esta característica resulta de gran valor para cualquier proyecto que se quiera desarrollar, puesto que brinda una mejor organización al código, y con ello, un excelente entendimiento del mismo. La Figura 9 muestra la estructura de carpetas de la propuesta de solución. La carpeta controller almacena las clases controladoras, con extensión js. La carpeta model guarda los archivos json, además de las clases que consumen el servicio leerJson. La carpeta view, recoge las clases de tipo html de la propuesta de solución.

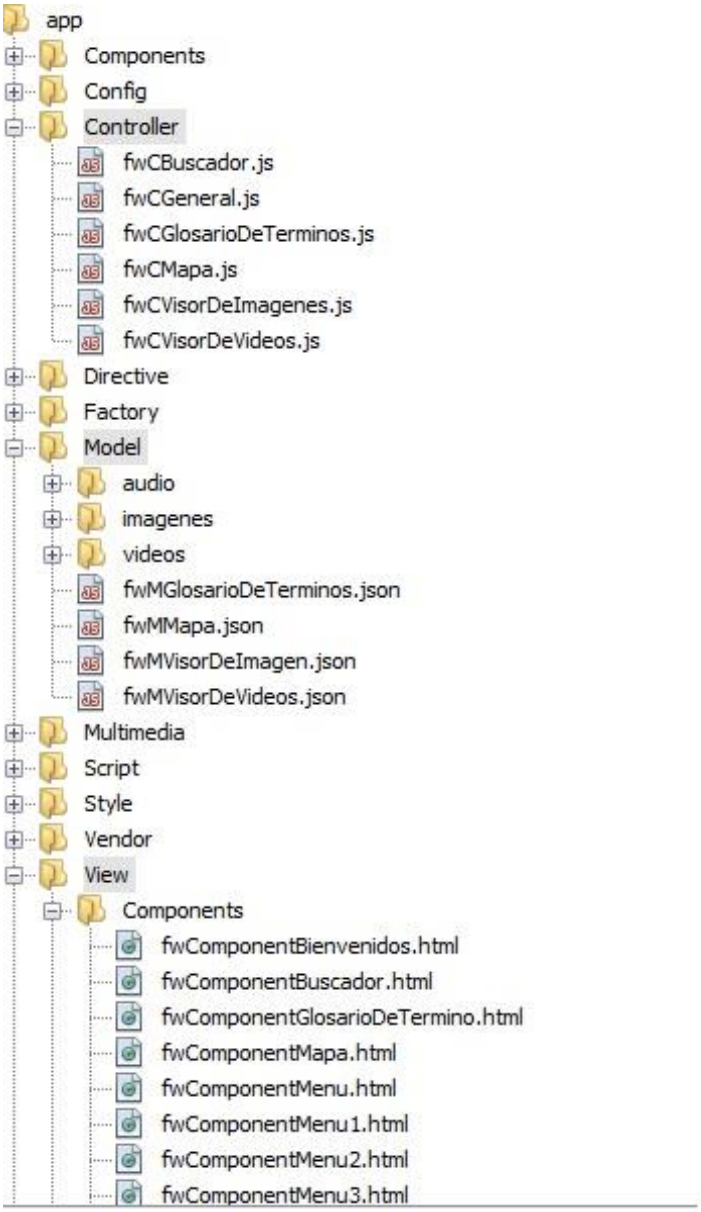


Figura 9: Estructura de carpetas de la propuesta de solución

3.3 Interfaz de usuario

Un prototipo es un artefacto que se genera con el objetivo de mostrar al cliente una visión rápida de lo que será la futura aplicación. Para su confección, el desarrollador conjuntamente con el cliente, define los requisitos que deberá cumplir un software determinado. Una vez obtenidos, se realiza un diseño rápido que lleva al prototipo que será mostrado al usuario, el cual evaluará y a partir de ahí se refinan los requisitos

levantados inicialmente (Pressman, 2005). El uso de prototipos puede provocar que el cliente se cree una falsa expectativa de lo que sería el *software* o que pida incorporar "pequeños cambios" que conviertan ese prototipo, en el resultado final. Esta condición conlleva frecuentemente a obtener una gestión de desarrollo de *software* muy lenta (Pressman, 2005). A pesar de estas desventajas, la construcción de prototipos puede ser un paradigma importante en la ingeniería de software (Pressman, 2005). El marco de trabajo con el cual será integrada la propuesta de solución aún no se encuentra disponible, razón que motivó a desarrollar un prototipo que brinde una visión del mismo.

Figura 10: Prototipo del marco de trabajo para desarrollar multimedia

El prototipo antes mostrado permitió confeccionar los prototipos de la propuesta de solución, los cuales se muestran a continuación:

Figura 11: Prototipo del componente Visor de imágenes

Figura 12: Prototipo del componente Visor de videos

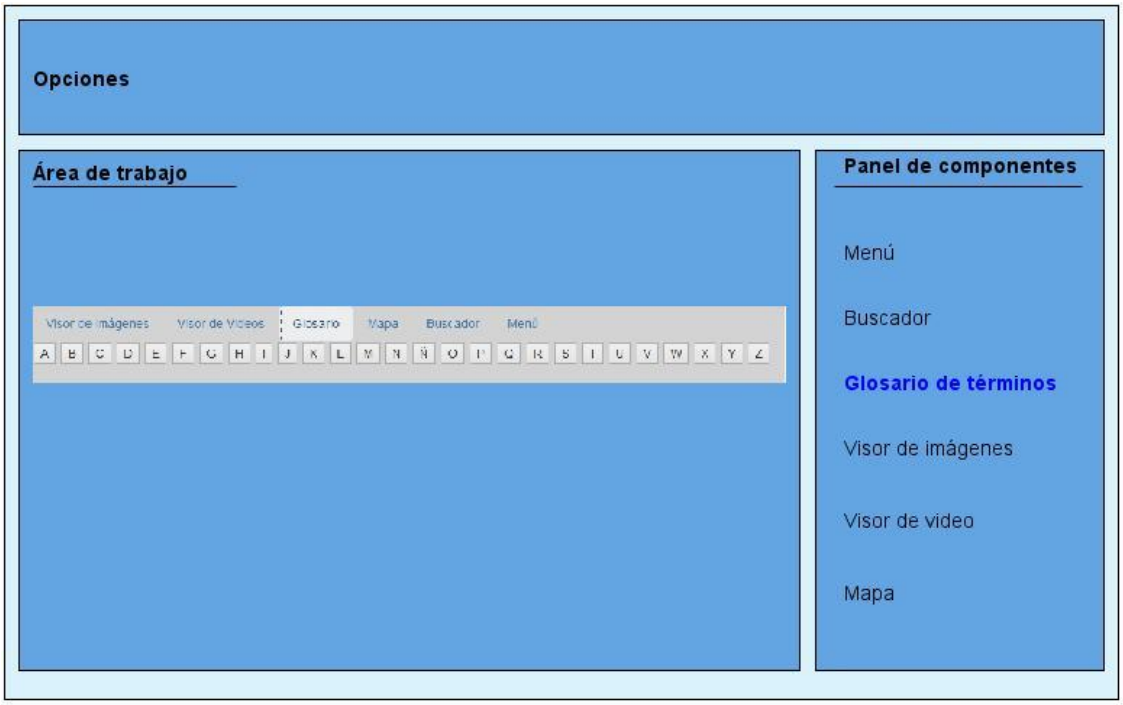


Figura 13: Prototipo del componente Glosario de términos

3.4 Estilos y estándares de codificación

Durante el proceso de implementación de un *software* es considerado una buena práctica realizar la codificación del mismo siguiendo estándares que guíen este proceso. Un código fuente legible permite una fácil navegación. Además, posibilita localizar y corregir errores de una manera más rápida, lo que conlleva a una mejor optimización del código (Osmani, 2012). Existen disímiles estilos y estándares de codificación, sin embargo, el desarrollador puede definir cómo desea que se visualice su código, es decir, puede determinar su manera de escribir y organizar el código fuente. Para el desarrollo de la propuesta de solución se utilizaron los estándares Idiomatic y CamelCase, definidos en el documento Estándares de codificación de la línea de multimedia a la que pertenece la presente investigación.

El estilo idiomatic.js recomienda cómo escribir código JavaScript. Además, es útil tanto para principiantes como para expertos. Entre los principales elementos que regula se encuentran los espacios en blanco, los paréntesis, las llaves, los fines de línea, las declaraciones, las funciones y las líneas vacías (Osmani, 2012). El estilo CamelCase estandariza el escribir palabras compuestas, donde la primera letra del identificador se escribe con minúscula y la primera letra de cada palabra concatenada se escribe con mayúscula (Marcos,

2013). Cuenta con dos variantes: UpperCamelCase y LowerCamelCase. El UpperCamelCase es cuando la primera letra de cada una de las palabras es mayúscula y el lowerCamelCase, es cuando la primera letra de la primera palabra es minúscula (Marcos, 2013). En la propuesta de solución se utilizará, el lowerCamelCase.

Utilizar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, constituyen factores de gran importancia para obtener un *software* con calidad y con buen rendimiento. Además, si se aplica de forma continuada un estándar de codificación bien definido, se utilizan técnicas de programación apropiadas, y se efectúan revisiones del código, aumentan las probabilidades de convertir un proyecto en un sistema de software fácil de comprender y mantener (Visual Studio.NET, 2003).

3.5 Pruebas

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

3.5.1 Desarrollo dirigido por pruebas (TDD)

Es una característica de la metodología AUP para comprobar el funcionamiento de los códigos que se van implementando. Este estilo sigue el principio "*test-first*", significa que las pruebas se crean antes de comenzar la implementación. "*Es un enfoque evolutivo para el desarrollo que combina el desarrollo de la prueba primero en el que se escribe una prueba antes de escribir el código de producción para cumplir con esa prueba"* (Beck, 2003). En TDD se escribe un nuevo código si una prueba automática ha fallado y se eliminan las duplicaciones (Beck, 2003).

3.5.2 Pruebas unitarias

Se encargan de probar una clase en concreto, testeando cada uno de sus métodos y viendo si dados unos parámetros de entrada, la salida es la esperada. La realización de estas pruebas debe consumir el menor tiempo posible, por lo que se hace necesario el uso de herramientas para automatizar este proceso.

El Jasmine es un marco de desarrollo impulsado por el comportamiento para probar el código JavaScript. No depende de otros *frameworks* de JavaScript. No requiere un DOM. Posee una sintaxis limpia, evidente

por lo que se puede escribir fácilmente pruebas. (Pivotal Labs, 2015). A continuación se muestran las principales funciones que ubican a este framework en la preferencia de los programadores (Pivotal Labs, 2015):

Tabla 5: Funciones de framework Jasmine para las pruebas unitarias.

Función	Descripción
describe(“nombre de la prueba”,**function**())	Función global que inicia las pruebas unitarias. Cuenta con dos parámetros: el nombre de la prueba, y una función que tiene el código que ejecutará la prueba.
specs	Representan afirmaciones ciertas o falsas en dependencia del resultado esperado con la prueba.
it(“nombre de la prueba”, **function**())	Función similar a **describe**, que especifica el método donde se realizará la prueba unitaria.
expect(valor actual).**matcher**(valor esperado)	Función que compara el valor actual con el valor esperado que almacena el **matcher**. A continuación se muestran los matcher comúnmente utilizados en las pruebas unitarias: • expect(X).toEqual (Y): verifica si los elementos X, Y son iguales. • expect(X).toBe (Y): verifica si los objetos X, Y son iguales. • expect(X).toBeNull (): verifica si el valor es nulo. • expect(X).toBeFalsy (): comprueba si el valor es falso. • expect(X).toBeTruthy (): verifica si el valor es verdadero. • expect(X).toContain (Y): verifica si el valor actual (X) contiene al valor esperado (Y). • expect(X).toBeUndefined (): comprueba si el valor es indefinido. • expect(X).toBeLessThan (Y): verifica si el valor actual (X) es menor que el valor esperado (Y). • expect(X).toBeGreaterThan (Y): verifica si el valor actual (X) es menor que el valor esperado (Y).

Luego de haber analizado las funciones que brinda Jasmine para realizar pruebas unitarias a un software, se muestra un ejemplo de cómo se desarrollaron estas pruebas en la propuesta de solución:

```
describe( 'Glosario de Terminos', function(){
    //Variables auxiliares
    var ctrl,
        leerJson = {
        url: {},
        getData: function(){ }
                  };

    beforeEach( module( 'multimedia' ) );

    beforeEach( inject( function( $rootScope, $controller, $q ){
        $scope = $rootScope.$new();
        spyOn( leerJson, 'getData' ).andReturn( $q.defer().promise );
        ctrl = $controller('glosario' ,{ $scope: $scope, leerJson: leerJson });

        $scope.datos = {data:[{
          "letra": "A",
            "palabras" : [
              {"p":"Amor", "des":"El amor nuca deja de ser"},
              {"p":"Antonio", "des":"Es mi amigo"},
              {"p":"Acuario", "des":"Lugar donde se crian pecesitos"}
            ]
        }]};
    }));

    it( 'Datos null', function (){
        expect( $scope.datos ).not.toBeNull();
    });

    it( 'mostrar palabras', function (){
        $scope.mostrar( 0 );
        expect( $scope.palabras ).not.toBeNull();
        expect( $scope.palabras.length ).toEqual( 3 );
    });
});
```

Figura 14: Pruebas unitarias realizadas al módulo Glosario de términos

3.6 Análisis de los resultados de las pruebas

Las pruebas unitarias fueron realizadas a todos los módulos que conforman la propuesta de solución, obteniendo en cada uno resultados favorables. La Figura 15 muestra los resultados que generaron las pruebas unitarias realizadas al módulo Visor de imágenes.

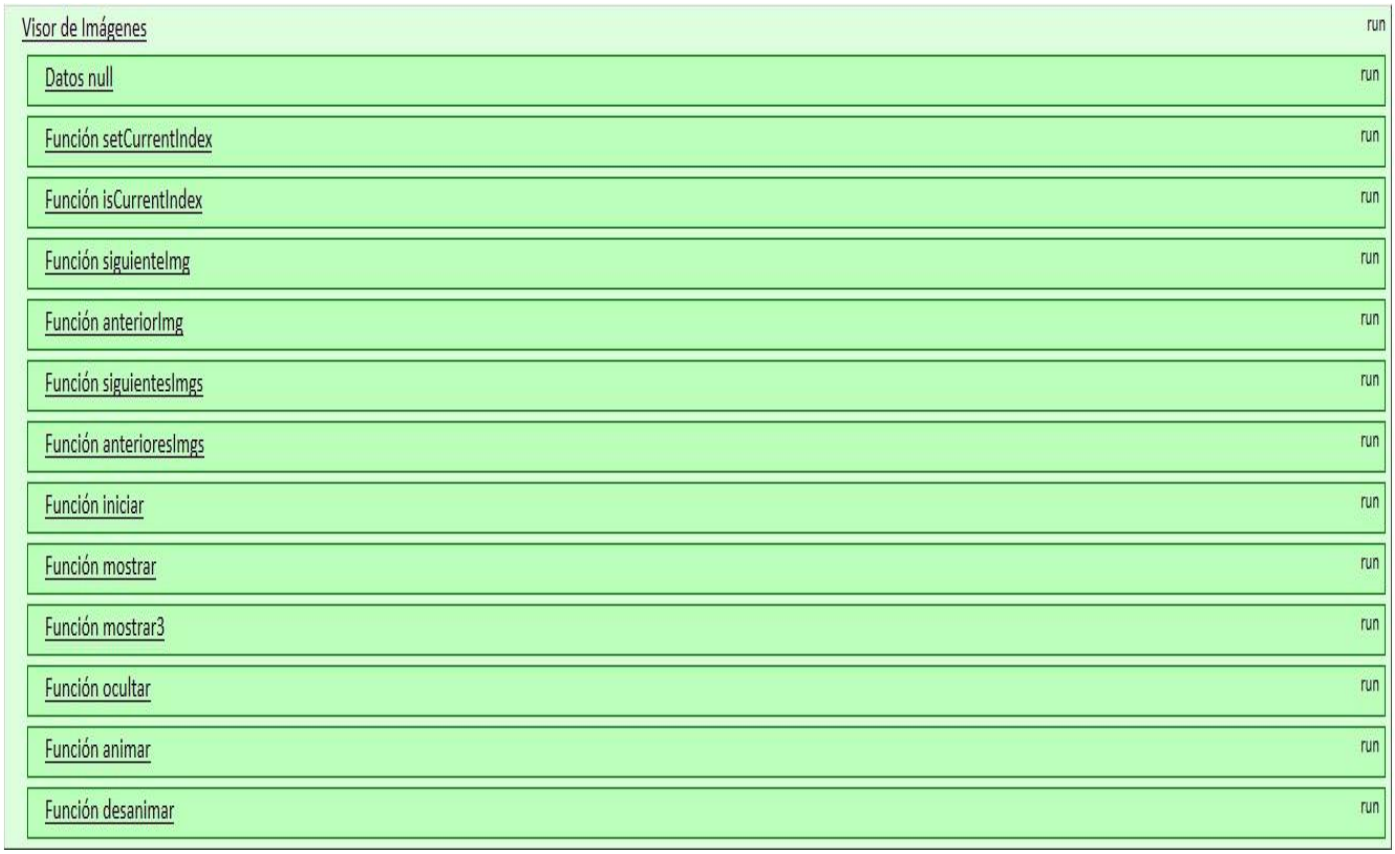


Figura 15: Resultados obtenidos en las pruebas unitarias. Módulo Visor de imágenes

Las pruebas realizadas al módulo Visor de imágenes generaron un total de 13 specs en 0.086s, donde ninguna fue fallida, lo cual representa un 100% de satisfacción. La Figura 16 muestra los resultados de las pruebas unitarias, que generó Jasmine para el módulo Visor de videos.

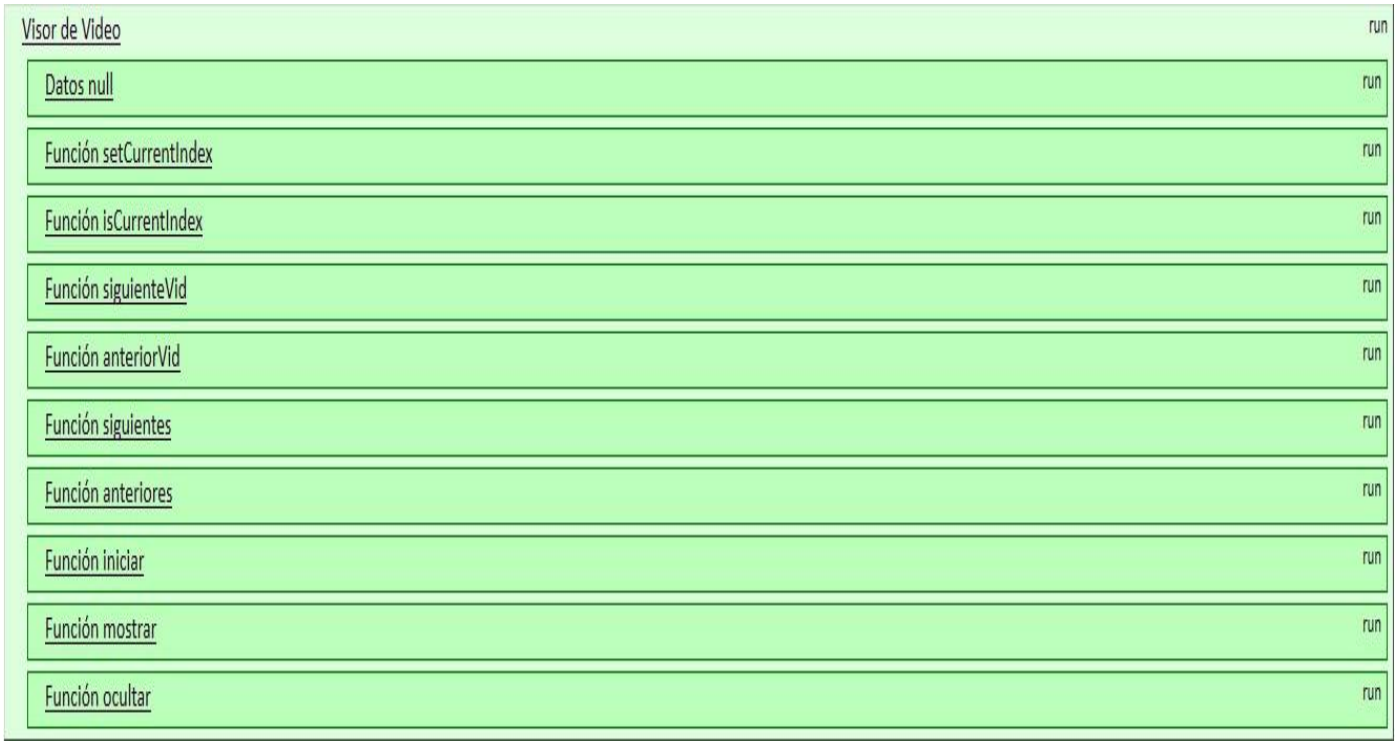


Figura 16: Resultados obtenidos en las pruebas unitarias. Módulo Visor de videos

Las pruebas realizadas al módulo Visor de videos arrojaron un total de 10 specs en 0.092s, donde todas fueron positivas, lo cual representa un 100% de satisfacción. La Figura 17 muestra los resultados obtenidos al realizar las pruebas unitarias al módulo Glosario de términos.

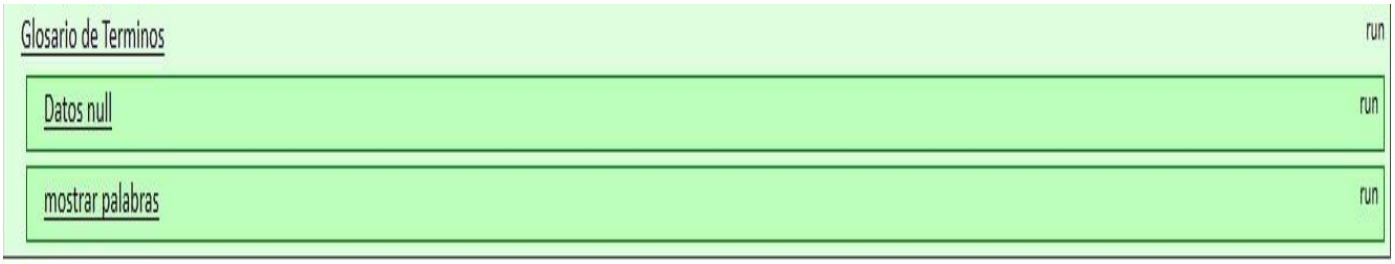


Figura 17: Resultados obtenidos en las pruebas unitarias. Módulo Glosario de términos

Fueron realizadas un total de 2 pruebas al módulo Glosario de términos en un tiempo de 0.086s. Ambas specs fueron satisfactorias, lo que significa un 100% de satisfacción.

Luego de analizar los resultados obtenidos en las pruebas unitarias, realizadas en cada uno de los módulos, se muestran los resultados generales de la validación de la propuesta de solución. La Figura 18 evidencia dichos resultados:

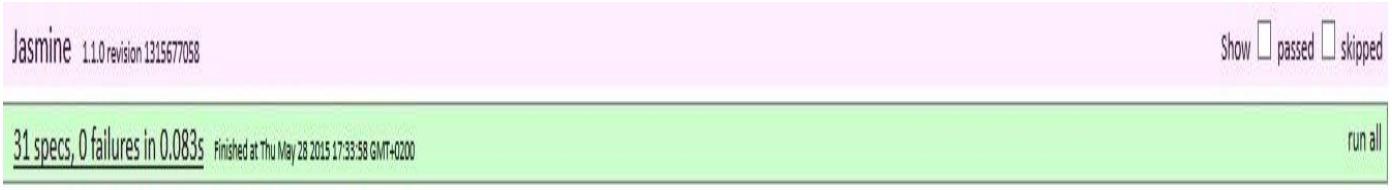


Figura 18: Resultados de las pruebas unitarias realizadas a la propuesta de solución

Fueron realizadas un total de 31 pruebas a la propuesta de solución, en un tiempo de 0.083s. No se obtuvieron resultados fallidos en las pruebas desarrolladas, lo que evidencia que la solución posee un alto nivel de calidad.

3.6 Conclusiones del capítulo

En este capítulo se describen las dos últimas disciplinas de la metodología de desarrollo de *software*, AUP: implementación y pruebas. Como parte del modelo de implementación se generó el diagrama de componentes. Este artefacto permitió obtener un mejor entendimiento de la relación de dependencia, uso e inclusión existente entre los componentes que conforman la propuesta de solución. El uso de los estándares de codificación Idiomatic y CamelCase unido a la estructura de carpetas definida por la arquitectura MVC, dotaron al código fuente de una mayor organización y claridad. Además, quedaron definidas las interfaces de usuario de la propuesta de solución, artefacto que brinda una visión de lo que será la aplicación. Fueron realizadas un total de 34 pruebas unitarias a los módulos de la propuesta de solución con el framework Jasmine, en un tiempo de 0.263s. No se obtuvieron resultados fallidos, lo que permitió determinar el alto nivel de calidad de la solución. El desarrollo de este capítulo mostró la importancia de aplicar buenas prácticas en la implementación de cualquier sistema informático. Además, evidenció la necesidad de realizar pruebas que permitan medir el grado de calidad de una aplicación, y con ello, conseguir la total satisfacción de las necesidades del cliente.

Conclusiones generales y Recomendaciones

Conclusiones generales

Al término de la investigación se arriba a las siguientes conclusiones, evidenciando el cumplimiento de los objetivos propuestos:

- El estudio del estado del arte mostró la existencia de insuficiencias en el proceso de desarrollo de multimedia en la universidad. Para contribuir con ello, se desarrollaron un total de seis componentes básicos, utilizando tecnologías y herramientas libres como son: NetBeans 8.0, AngularJS, JavaScript, HTML5 y CSS3.
- Se obtuvieron los artefactos de ingeniería necesarios para la implementación, entre los que se incluye:
  - El modelo conceptual con 10 conceptos identificados.
  - El listado de 14 requisitos funcionales y 10 no funcionales.
  - El modelo de clases del diseño con 6 diagramas de clases realizados.
  - El modelo de implementación con un diagrama de componentes realizado.
- El desarrollo de los componentes básicos permitió mejorar el proceso de creación de multimedia en la universidad y su integración al marco de trabajo de desarrollo de multimedia permitirá:
  - Desarrollar multimedia con tecnologías libres.
  - Obtener una herramienta capaz de brindar un ambiente de desarrollo multiplataforma.
- Fueron realizadas 34 pruebas unitarias al código fuente de la propuesta de solución, obteniéndose un 100% de satisfacción.

Recomendaciones

Tomando como base la investigación realizada y el análisis de los resultados obtenidos, se recomienda:

- Integrar los componentes básicos obtenidos durante la presente investigación al marco de trabajo de desarrollo de multimedia con tecnologías libres.
- Agregar un tema en blanco a cada uno de los componentes para que sean personalizables.

Referencias bibliográficas

1. OSMANI, Addy, 2012, *Learning JavaScript Design Patterns*. O'Reilly Media, Inc. ISBN 9781449334871.

2. ALEGSA, [sin fecha], ¿Cuál es la definición de Arquitectura? [En línea]. [Accedido 16 Mazo 2015]. Disponible en: http://www.alegsa.com.ar/Dic/arquitectura.php
Alegsa.com.ar

3. ÁLVAREZ CAULES, Cecilio, 2012, *ARQUITECTURA JAVA SÓLIDA*. ISBN 978-1-291-16766-5. Certificado de Empresa Architect

4. BECK Kent, 2003, *Test-driven Development: By Example*. Addison-Wesley Professional. ISBN 9780321146533.

5. BELLOCH, Consuelo, 2012, *APLICACIONES MULTIMEDIA*. Universidad de Tecnología Educativa (UTE). Universidad de Valencia.

6. Blanco, Angel Fidalgo. *Multimedia educativa.* Universidad Politécnica de Madrid : Laboratorio de Innovación en Tecnologías de la Información.

7. BOOCH Grady, JACOBSON Ivar and RUMBAUGH James, 2000, *EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE*. Madrid: Addison-Wesley. ISBN 84-7829-036-2

8. BOOTSTRAP, 2015, Bootstrap. [En línea]. 2015. [Accedido 12 Junio 2015]. Disponible en: http://getbootstrap.com/

9. CABRERO ALMENARA, Julio and DUARTE HUEROS, Ana, [sin fecha], Evaluación de medios y materiales de enseñanza en soporta multimedia. [En línea]. [Accedido 11 Febrero 2015]. Disponible en: http://www.sav.us.es/pixelbit/pixelbit/articulos/n13/n13art/art133.htm

10. CHAFFER, Jonathan and SWEDBERG, Karl, 2010, *JQuery Reference Guide: A Comprehensive Exploration of the Popular JavaScript Library*. Packt Publishing Ltd. ISBN 9781849510059.

11. CUEVAS, Ignacio Aedo. *Sistemas multimedia: Análisi, diseño y evaluación.* Madrid : Universidad Nacinoal de Educación a Distancia, 2009.

12. DARWIN, Peter Bacon and KOZLOWSKI Pawel, 2013, *Mastering Web Application Development with AngularJS* [En línea]. Packt Publishing. [Accedido 25 Marzo 2015]. ISBN 978-1-78216-183-7. Disponible en: http://proquest.tech.safaribooksonline.de/9781782161820

13. FLANAGAN David, 2007, *JavaScript. La Guía Definitiva* [En línea]. España. [Accedido 5 Febrero 2015]. ISBN 978-84-415-2202-2 84-415-2202-2. Disponible en: http://dialnet.unirioja.es/servlet/libro?codigo=316551

14. GARCIA, NICADO Miriam. *Política de migración .* RESOLUCIÓN No. 179/14, La Habana : Dirección de servicios jurídicos, 2014.

15. GAMMA Erich, HELM Richard, JONHSON Ralph and VLISSIDES John, 1994, *Dessign Patterns: Elements of Reusables Object-Oriented Software*. Addison-Wesley. ISBN 0-201-63361-2.

16. GAUCHAT, Juan Diego, 2012, *El gran libro de HTML5, CSS3 y Javascript.* Marcombo. ISBN 978-84-267-1782-5.

17. GONZALEZ, LLERGO Pedro Emmanuel, 2011, Modelo-Vista-Controlador. *Google Docs* [En línea]. 2 Octubre 2011. [Accedido 16 Marzo 2015]. Disponible en: https://docs.google.com/document/d/1iKogJyDMCUi_NVxbIcOXIb3B5f_1HPzoK8V3fqTXC7g/edit?hl=es&pli=1&usp=embed_facebook

18. GREEN, Brad and SESHADRI, Shyam, 2013, *AngularJS*. O'Reilly Media, Inc. ISBN 9781449355883.

19. HERNÁNDEZ LEÓN, Rolando Alfredo and COELLO GONZÁLEZ, Sayda, 2002, *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA*. Cuidad de la Habana: EDUNIV. ISBN 959-16-0343-6. Universidad de las Ciencias Informáticas.

20. IEEE, 2012, *Latin America Transactions, IEEE (Revista IEEE America Latina)* [En línea]. 16 Enero 2012. [Accedido 24 Marzo 2015]. Disponible en:

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6129710&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6129710

21. IMELDA LATAPIE VENEGAS, [Sin fecha], *Acercamiento al aprendizaje multimedia* [En línea]. [Accedido 6 Febrero 2015]. Disponible en: http://scholar.google.com.cu/scholar_url?url=http%3A%2F%2Fdialnet.unirioja.es%2Fdescarga%2Farticulo%2F2695335.pdf&hl=es&sa=T&oi=ggp&ct=res&cd=7&ei=sv7UVKuoMI_6qAGNv4D4Ag&scisig=AAGBfm1VIjaOQ6M3lUo0aZrfyeGflS7GRg&nossl=1&ws=1366x643 Universidad Simón Bolívar, Universidad Autónoma Metropolitana.

22. LANCKER, Luc Van, 2012, *jQuery: El framework JavaScript de la Web 2.0*. Ed. ENI. ISBN 9782746072589.

23. LANCKER, Luc Van, 2013, *HTML5 y CSS3: Domine los estándares de las aplicaciones Web [2ª edición]*. Ediciones ENI. ISBN 9782746084063.

24. LARMAN, Craig, 2003, *UML Y PATRONES Introducción al análisis y diseño orientado a objetos y al proceso unificado.* Segunda. Madrid: Prentice Hall. ISBN 8420534382.

25. Marcos, Pedro, 2013. Pedromarcos.com. *Escritura estilo CamelCase.* [En línea] [Accedido 10 mayo 2015]. http://www.pedrormarcos.com/escritura-estilo-camelcase/.

26. MARTÍN, Alfonso Gutiérrez, 1997, *Educación multimedia y nuevas tecnologías*. Ediciones de la Torre. ISBN 9788479604554.

27. MELLADO DOMÍNGUEZ, Javier, 2008, *Ajax* [en línea]. España. [Accedido 12 Junio 2015]. ISBN 978-84-415-2414-9. Available from: http://dialnet.unirioja.es/servlet/libro?codigo=313653
Editores: Anaya Multimedia

28. MICROSOFT, 2005, Dessign Patterns: Dependency Injection. [En línea]. 2005. [Accedido 1 Abril 2015]. Disponible en: https://msdn.microsoft.com/en-us/magazine/cc163739.aspx

29. VISUAL STUDIO.NET, 2003, Revisiones de código y estándares de codificación. [En línea]. 2003. [Accedido 13 Junio 2015]. Disponible en: https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx

30. ORACLE CORPORATION©, 2015, Bienvenido a NetBeans. *NetBeans IDE* [En línea]. 2015. [Accedido 10 Febrero 2015]. Disponible en: https://netbeans.org/index.html.

31. PINTO, María. ALFAMEDIA. [En línea]. 13 Abril 2011. [Accedido 3 Febrero 2015]. Disponible en: http://www.mariapinto.es/alfamedia/cultura/entornos.htm

32. PIVOTAL LABS, 2015, Jasmine: Behavior-Driven JavaScript. [En línea]. [Accedido 13 Junio 2015]. Disponible en: http://jasmine.github.io/

33. PRESSMAN, Roger S. Ingeniería del software: un enfoque práctico. Séptima Edición. Mcgraw Hill/Interamericana Editores, 2005. ISBN 9789701054734.

34. RONDÓN Cedeño, Sandy Orlando, 2007, *Multimedia sobre la herramienta Macromedia Fireworks.* Cuidad de la Habana: Universidad de las Ciencias Informáticas. Tutor: Ing. Guillermo Solenzal Fernández.

35. RODRÍGUEZ SÁNCHEZ, Tamara, 2014, *Metodología de desarrollo para la Actividad productiva de la UCI*. 12 November 2014. Programa de mejora. Universidad de las Ciencias Informáticas.

36. SÁNCHEZ ALONSO, Salvador, SICILIA URBÁN, Miguel Ángel and RODRÍGUEZ GARCÍA, Daniel, 2012, *Ingeniería del Software: Un enfoque desde la guía SWEBOK*. Primera Edición. Departamento de Ciencias de la Computación. Universidad de Alcalá: Alfaomega, Grupo Editor, México. ISBN 9786077074205.

37. VALBUENA APONTE, Ángela María, 2014, *Guía comparativa de Frameworks para los lenguajes HTML 5, CSS y JavaScript para el desarrollo de aplicaciones Web* [En línea]. Tesis. Universidad Tecnológica de Pereira. [Accedido 1 Abril 2015]. Disponible en: http://repositorio.utp.edu.co/dspace/handle/11059/4577

38. VISUAL PARADIGM©. Download UML, BPMN modeling tools with project management. In: [En línea]. [Accedido 10 febrero 2015]. Disponible en: http://www.visual-paradigm.com/download/.

40. VISUAL PARADIGM©. Visual modeling tool for building enterprise applications. In: [En línea]. [Accedido 10 Febrero 2015]. Disponible en: http://www.visual-paradigm.com/product/vpuml/features/.

Anexos

Descripción textual de requisitos funcionales.

Tabla 6: Descripción del requisito Listar temas de galería de imágenes

Precondiciones	Deben existir imágenes para visualizar en la galería.	
Flujo de eventos		
Flujo básico <<Listar temas de galería de imágenes>>		
1.	El desarrollador selecciona el componente visor de imágenes.	
2.	Se muestran los cuatro temas disponibles para el visor de imágenes.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran los cuatro temas predefinidos del visor de imágenes.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
2.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan las imágenes subidas	
Conceptos	**Galería de imágenes**	Componente que permite visualizar imágenes.
	Imágenes	Título: Palabra o conjunto de palabras que dan información sobre la imagen. Descripción: Breve comentario sobre la imagen. Formato: Forma de escritura de la imagen “png”. Tamaño: Formato de la imagen 800x600. Dirección: Lugar donde se encuentra la imagen.
Requisitos especiales	Se debe tener en cuenta el tamaño de la imagen y el formato de la misma.	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable.	

Tabla 7: Descripción del requisito Seleccionar tema de galería de imágenes.

Precondiciones	Deben existir imágenes para visualizar en la galería. Debe haberse ejecutado el requisito funcional: Listar temas de galerías de imágenes.	
Flujo de eventos		
Flujo básico <<Seleccionar tema de galería de imágenes>>		
1.	El desarrollador selecciona el tema de galería de imágenes que desee.	

2.	Se muestran el tema seleccionado.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran las imágenes según el tema seleccionado.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
2.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan las imágenes subidas.	
Conceptos	**Galería de imágenes**	Componente que permite visualizar imágenes.
	Imagen	Título: Palabra o conjunto de palabras que dan información sobre la imagen. Descripción: Breve comentario sobre la imagen. Formato: Forma de escritura de la imagen “png”. Tamaño: Formato de la imagen 800x600. Dirección: Lugar donde se encuentra la imagen.
Requisitos especiales	Se debe tener en cuenta el tamaño de la imagen y el formato de la misma	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable.	

Tabla 8: Descripción del requisito Añadir descripción a la imagen.

Precondiciones	Deben existir imágenes para visualizar en la galería. Debe haberse ejecutado el requisito funcional: Listar temas de galerías de imágenes. Debe haberse ejecutado el requisito funcional: Seleccionar tema de galería de imágenes.
Flujo de eventos	
Flujo básico <<Adicionar descripción a la imagen>>	
1.	El desarrollador selecciona el archivo json correspondiente a las imágenes y modifica el atributo descripción.
2.	Se muestra la descripción de la imagen.
Pos-condiciones	
1.	Se muestra la descripción de la imagen.
Flujos alternativos	

Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
2.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan las imágenes subidas y la descripción	
Conceptos	**Galería de imágenes**	Componente que permite visualizar imágenes.
	Imagen	Título: Palabra o conjunto de palabras que dan información sobre la imagen. Descripción: Breve comentario sobre la imagen. Formato: Forma de escritura de la imagen “png”. Tamaño: Formato de la imagen 800x600. Dirección: Lugar donde se encuentra la imagen.
Requisitos especiales	Se debe tener en cuenta el tamaño de la imagen y el formato de la misma.	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable. Insertar una descripción a una imagen sin tener que acceder al archivo json.	

Tabla 9: Descripción del requisito Ampliar imagen.

Precondiciones	Deben existir imágenes para visualizar en la galería. Debe haberse ejecutado el requisito funcional: Listar temas de galerías de imágenes. Debe haberse ejecutado el requisito funcional: Seleccionar tema de galería de imágenes.
Flujo de eventos	
Flujo básico <<Ampliar imagen>>	
1.	El desarrollador selecciona la imagen que desee visualizar.
2.	Se muestra la imagen ampliada.
Pos-condiciones	
1.	Se ejecuta la funcionalidad y se muestra la imagen ampliada.
Flujos alternativos	
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>	
1.	
Pos-condiciones	
1.	
Validaciones	

2.		
Conceptos	**Galería de imágenes**	Componente que permite visualizar imágenes.
	Imagen	Título: Palabra o conjunto de palabras que dan información sobre la imagen. Descripción: Breve comentario sobre la imagen. Formato: Forma de escritura de la imagen “png”. Tamaño: Formato de la imagen 800x600. Dirección: Lugar donde se encuentra la imagen.
Requisitos especiales	Se debe tener en cuenta el tamaño de la imagen y el formato de la misma.	
Asuntos pendientes	Añadir un tema en blanco al componente, para que pueda ser configurable y personalizable.	

Tabla 10: Descripción del requisito Listar temas de menús.

Precondiciones		
Flujo de eventos		
Flujo básico <<Listar temas menús>>		
1.	El desarrollador selecciona el componente menú.	
2.	Se muestran los temas disponibles para visualizar el menú.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran los cuatro temas predefinidos para el componente menú.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan las rutas de acceso del menú	
Conceptos	**Menús**	Componente que define cuatro temas diferentes de visualizar los restantes componentes.
Requisitos especiales	El link debe tener relación con los creados en la configuración del módulo.	
Asuntos pendientes	Añadir más temas al componente menú y agregar el tema en blanco para lograr que sea personalizable y configurable.	

Tabla 11: Descripción del requisito Seleccionar temas de menú

Precondiciones	Debe haberse ejecutado el requisito funcional: Listar temas de menús.
Flujo de eventos	
Flujo básico <<Seleccionar tema de menú>>	
1.	El desarrollador selecciona el tema del componente menú que desee.

2.	Se muestra el tema seleccionado.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestra el menú de acuerdo al tema seleccionado.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
Pos-condiciones		
1.		
Validaciones		
1.	Se validan las rutas del menú	
Conceptos	**Menús**	Componente que define cuatro temas diferentes de visualizar los restantes componentes.
Requisitos especiales	El link debe tener relación con los creados en la configuración del módulo.	
Asuntos pendientes	Añadir más temas al componente menú y agregar el tema en blanco para lograr que sea personalizable y configurable.	

Tabla 12: Descripción del requisito Realizar búsquedas

Precondiciones		
Flujo de eventos		
Flujo básico <<Realizar búsqueda de información>>		
1.	El desarrollador selecciona el componente buscador.	
2.	Introduce una palabra clave y da clic en el link buscar.	
3.		
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran los archivos json que incluyen la palabra introducida.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
Pos-condiciones		
1.		
Validaciones		
1.	Se valida el término escrito en el buscador.	
Conceptos	**Buscador**	Componente que permite realizar búsquedas en la multimedia.
Requisitos especiales	Se debe tener en cuenta las tildes en las palabras.	
Asuntos pendientes	Añadir un tema en blanco al componente para lograr que sea personalizable y configurable.	

Tabla 13: Descripción del requisito Mostrar glosario de términos

Precondiciones		
Flujo de eventos		
Flujo básico <<Mostrar glosario de términos>>		
1.	El desarrollador selecciona el componente glosario de términos.	
2.	Se muestra el abecedario.	
3.	El desarrollador selecciona una letra.	
Pos-condiciones		
1.	Se ejecuta la funcionalidad y se muestran todas las palabras que comienzan con la letra seleccionada y su significado.	
Flujos alternativos		
Flujo alternativo <<Nº Evento>>.<<letra iniciando por la a>> <Condición que dio lugar a la extensión>		
1.		
Pos-condiciones		
1.		
Validaciones		
1.	Se valida la letra seleccionada.	
Conceptos	**Glosario**	Componente que permite conocer el significado de una palabra, luego de haber seleccionado una letra.
Asuntos pendientes.	Añadir un tema en blanco al componente para lograr que sea personalizable y configurable.	

Diagramas de clases del diseño

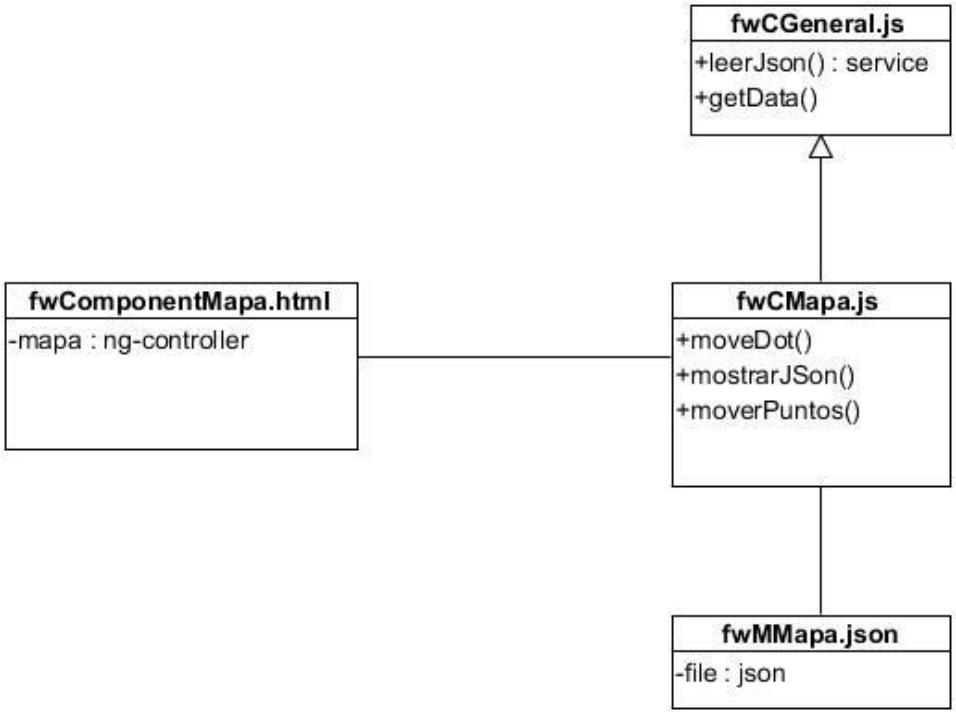


Figura 19: Diagrama de clases del diseño. Componente Trabajo con mapas

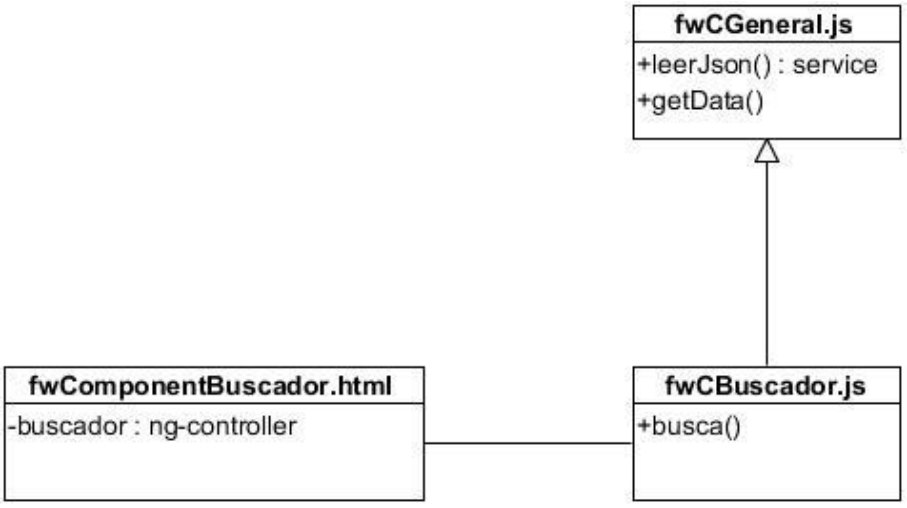


Figura 20: Diagrama de clases del diseño. Componente Buscador.

Prototipos de interfaz de usuario

Figura 21: Prototipo del componente Mapa

Figura 22: Prototipo del componente Buscador