FACULTAD 8

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Colección de Corales Pétreos

Autores: Nilber Barbán Góngora

Sandy Alberto Gallardo

Tutores: Ing. Guillermo Solenzal Fernández

Ing. Mailyn Cabrera Torres

Ciudad de la Habana, junio 2009

Declaración de Autoría

Declaramos ser los autores de la presente tesis y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso de ella para lo que necesite, cediéndole de esta forma los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autores: Nilber Barbán Góngora Sandy Alberto Gallardo

_____ _____

Tutores: Ing. Guillermo Solenzal Fernández Ing. Mailyn Cabrera Torres

_____ _____

“Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad.”

Albert Einstein

De Nilber:

A mi mamá y mi papá, por ser ellos mi razón de ser, por la educación, la comprensión, el apoyo y el amor que siempre me han dado.

De Sandy:

Dedico este trabajo a mis padres, especialmente a mi mamá por toda la ayuda y el apoyo que me ha brindado en estos años, en cada momento difícil, en vísperas de cada examen, en cada tarea y en cada instante, y mi papá por haberme inculcado esa gran fuerza de voluntad por el estudio y por enseñarme a no rendirme nunca. A ellos dedico este trabajo.

De Nilber:

Mi más grande agradecimiento es para mi mamá y mi papá, por haber confiado siempre en mí y en mis decisiones, por llevarme en su corazón. Solo quiero que estén orgullosos de su “niño”.

A mi hermano, por ser mi ejemplo y estar para mí cuando lo he necesitado, tú siempre serás “niño gande”. Al resto de mi familia, por estar siempre al tanto de mi desarrollo.

A Pablo, Yunior y Jesús, por toda la ayuda brindada. Después de todo los _level si funcionan.

A Yailyn, mi novia, por toda la paciencia y comprensión. Desde ahora soy Ing. Nilber Barbán Góngora.

A Mailyn y Guillermo, mis tutores, por su profesionalidad y su ayuda. Mailyn ya no me debes nada.

A mis compañeros de grupo que han venido conmigo durante estos cinco años, los cuales han sido maravillosos. Siempre tendrán un lugar en mi corazón.

A todos mis amigos de la UCI y del barrio, por todo su apoyo.

A mi compañero de tesis, Sandy, por su buen trabajo en el diseño.

A los especialistas del Acuario Nacional Cuba, por tener que soportarnos todo este tiempo.

A todos los que han tenido que ver con mi formación desde mis primeros años de vida hasta el día de hoy.

De Sandy:

A mi mamá, por ser ella la razón de todo mi ser, por todo su cariño y amor, por ser ella mi luz y mi ejemplo a seguir.

A mi papá, por haberme ayudado en todo momento, por todas sus enseñanzas y por todo su apoyo.

A mi familia, a mis tíos, tías y primos, a todos ellos gracias por haberme apoyado siempre.

A mi hermana, que aunque siempre ha estado lejos, se que desde dónde estás nunca has dejado de pensar en mí.

A Emelina y a toda su familia, por ser mi segunda familia.

A Yovana, a Carlos, a Ruly, a Dagoberto gracias por su ayuda.

A todos mis amigos de la UCI y especialmente a mis compañeros de grupo, gracias por todos los buenos momentos y también por el apoyo en los malos.

A mis amigos de Fomento, Yusmar, Gustavo, Robe, Yosvany, Rolando, Oslando, Diasmel, Rafael, Alexei, Yenier, Ainel a todos muchas gracias.

A mi compañero de tesis Nilber por su dedicación y preocupación.

A mis compañeros del Proyecto MENPET, Pablo, Yunior, Jesús, a los profes Yorgelys, Sergio y a mis tutores Guillermo y Mailyn por todo su apoyo y su preocupación por el desarrollo de este trabajo.

RESUMEN

El tema central del presente trabajo es el desarrollo de la aplicación con tecnología multimedia Colección Corales Pétreos. Se realiza un estudio de las tendencias actuales en el desarrollo de este tipo de aplicaciones, además se hace el análisis y selección de las herramientas y tecnologías más adecuadas para este particular. Se exponen los resultados de la investigación realizada sobre las metodologías de desarrollo de software, los lenguajes de modelado, las herramientas de autor, así como los lenguajes de programación y se justifican las selecciones hechas para el desarrollo de la aplicación. Para concluir se lleva a cabo la implementación del software a partir de los artefactos obtenidos del análisis y el diseño del mismo.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Entre los años 1970 y 1973 comienza a formarse la Colección de Corales Pétreos de Cuba en el Instituto de Oceanología de la entonces Academia de Ciencias de Cuba, hoy CITMA. En esta fecha se inicia el estudio detallado de la fauna de corales pétreos en los mares cubanos llevado a cabo por los investigadores Vassil Zlatarski de la Academia de Ciencias de Bulgaria y Nereida Martínez-Estalella del Instituto de Oceanología de Cuba. Esta colección cuenta con 4,980 ejemplares reunidos en 15 familias, 28 géneros, 41 especies, 4 subespecies y 23 formas, de estas, 5 nuevas para la ciencia. Actualmente se conserva en el Acuario Nacional de Cuba y está considerada como una de las más completas y mejor conservadas en instituciones similares del área de América Latina y el Caribe. Prueba de ello fue el inventario realizado en el año 2003, el cual mostró que el 83% de la colección se conserva en perfecto estado.

La colección ha viajado dos veces a Bulgaria. Para su traslado fue empaquetada en 22 cajas de madera de grandes dimensiones (220 cm x 120 cm x 100 cm) donde cada ejemplar permanecía envuelto en papel para protegerlo de golpes. La primera vez que viajó fue en 1975, por vía aérea desde La Habana hasta Sofía, Bulgaria, donde se concluyó el trabajo de laboratorio y el manuscrito de la monografía. La colección permaneció dos años en la Academia de Ciencias de Bulgaria donde cada ejemplar fue fotografiado por un fotógrafo búlgaro y filmada en un documental que nunca llegó a Cuba, después de este tiempo regresó a la Habana.

La colección se mantuvo en el Instituto de Oceanología (IDO) hasta el 2005, en ese año la dirección de CITMA decidió que todas las colecciones del IDO pasaran al Acuario Nacional de Cuba (ANC) y una vez más la colección de corales fue empaquetada pieza por pieza, cada una de ellas envuelta en papel y colocada en cajas esta vez de cartón de muchos tamaños. Desde esa fecha permanece en el Departamento de Colecciones Marinas del ANC.

En la actualidad numerosos especialistas de diversos países del mundo consultan esta colección, sus ejemplares han sido estudiados en varios cursos de postgrado de taxonomía, impartido en ocasiones por los formadores de la colección, a especialistas cubanos y extranjeros.

Aunque el por ciento de conservación de la colección está considerado entre los mejores del mundo, su carácter móvil para su estudio y/o exposición, ha provocado deterioros que se reflejan en su estado actual,

de aquí la importancia de buscar una alternativa para permitir el estudio de estos ejemplares sin que sean trasladados. Actualmente no existe un producto informático que permita a investigadores e interesados en el tema, estudiar la Colección de Corales Pétreos existentes en nuestro país, por lo que se decide definir éste como problema científico de nuestra investigación.

Con el propósito de encontrar una solución al problema planteado se define como objetivo general: desarrollar una aplicación con tecnología multimedia que permita digitalizar la colección y facilitar su estudio desde diferentes lugares del mundo, objetivo que de cumplirse con éxito ayudaría además a reducir en un alto por ciento el deterioro de la colección. Como objetivos específicos se plantean los siguientes:

1. Realizar el estudio del estado del arte del objeto de la investigación científica.
2. Realizar análisis y diseño del producto.
3. Implementar el producto.

A partir del problema planteado se decide estudiar el proceso de desarrollo de software con tecnología multimedia, y como campo de acción específicamente el proceso de análisis, diseño e implementación del software con tecnología multimedia “Colección de Corales Pétreos”. La idea a defender será que si se logra desarrollar un producto informático que use tecnología multimedia, entonces se podrán realizar estudios de la colección de corales pétreos existente en Cuba sin necesidad de trasladarlos de su lugar de conservación, prolongando así los años de vida de dicha colección.

Para dar cumplimiento a los objetivos específicos y al objetivo general se han propuesto las siguientes tareas de investigación:

1. Realizar entrevistas a los clientes y especialistas en el tema para determinar el diseño de interfaz gráfico.
2. Estudiar e identificar las tecnologías existentes en Cuba y en el resto del mundo para el desarrollo del producto.
3. Estudiar las metodologías y herramientas para la confección de los artefactos del sistema durante el desarrollo del software.

4. Realizar el levantamiento de los requisitos y medias a utilizar como contenido del producto informático.
5. Generar los artefactos correspondientes al diseño del producto.
6. Desarrollar el diseño de interfaz gráfica.
7. Realizar implementación de la aplicación.
8. Elaboración del documento de tesis de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El presente capítulo tiene entre sus objetivos hacer una evaluación del estado en que se encuentra actualmente la tecnología multimedia, haciendo referencia a conceptos generales asociados a ella. Conjuntamente se analizarán diferentes metodologías, lenguajes y herramientas de desarrollo de software para definir, a partir de la comparación, cuáles de ellas se utilizarán para dar solución al problema planteado.

1.2 Tecnología Multimedia

Existen diferentes conceptos asociados al término tecnología, la Real Academia Española lo define como:

> **1.** f. Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico. [1]

Al hablar de multimedia no se hace referencia a un nuevo concepto, este existe desde antes que surgieran términos tales como producto electrónico o aplicaciones informáticas, y se resume en:

> **1.** adj. Que utiliza conjunta y simultáneamente diversos medios, como imágenes, sonidos y texto en la transmisión de una información. [2]

La tecnología multimedia tiene su antecedente más remoto en dos vertientes: el invento del transistor con los desarrollos electrónicos que propició, y los ejercicios eficientes de la comunicación, que buscaba asegurar la recepción del mensaje y su correcta percepción mediante la redundancia.

A inicios de los años 50 el invento del transistor posibilitó la revolución de la computadora, con la fabricación del chip, los circuitos eléctricos y las tarjetas electrónicas, los cuales dieron lugar a las unidades compactas de procesamiento y la integración del video. Todo esto, junto con los desarrollos de discos duros, flexibles y discos ópticos, se ha concretado en la tecnología de las computadoras personales. Posteriormente, fueron desarrollados una serie de accesorios y periféricos para que la computadora pueda manejar imagen, sonido, gráficos y videos, además del texto.

El 24 de enero de 1984 es lanzado por Apple Inc. el primer ordenador personal comercializado exitosamente, el Macintosh. Este usaba una interfaz gráfica de usuario y un mouse en vez del estándar de esa época: la interfaz por línea de comandos. Conjuntamente con su capacidad de reproducción de

sonido comparable con uno de amplitud modulada, su sistema operativo y demás programas desarrollados dieron lugar a la primera posibilidad de lo que se conoce como Multimedia.

La tecnología multimedia toma auge en los video-juegos, a partir de 1992, cuando se integran: audio (música, sonido estéreo y voz), video, gráficos, animación y texto al mismo tiempo. La principal idea multimedia desarrollada en estos es: que se pueda navegar y buscar la información que se desea sobre un tema, sin tener que recorrer todo el programa, que se pueda interactuar con la computadora y que la información no sea lineal sino asociativa.

En términos generales, podemos hablar de diversos niveles de difusión de las aplicaciones con tecnología multimedia. Las desarrolladas por las empresas conciernen a tres niveles principales: la formación, la comercialización y las comunicaciones. Además existen cuatro grupos importantes orientadas al consumidor individual: las aplicaciones centradas en la computadora, las redes de comunicación (incluyendo Internet y servicios diversos de telecomunicación) y en industria del entretenimiento (la televisión y los videojuegos).

Las aplicaciones con tecnología multimedia deben cumplir con principios tales como:

- Principio de la necesidad: el producto debe servir para algo.
- Principio de la atención: el producto debe conseguir que el usuario mantenga una actitud continua de expectación ante el audiovisual.
- Principio de la economía:
    - Economía de tiempo: evitar secuencias demasiado largas. En lenguaje audiovisual la mínima demora es mucho tiempo perdido.
    - Economía de espacio: ubicar las medias necesarias y con el tamaño necesario.
    - Economía conceptual: los textos que acompañan al resto de las medias no deben sobre informar a los usuarios. Debe permitírsele al usuario pensar.
    - Economía de lenguaje: no se debe ser demasiado exhaustivo e incluir cada punto o coma que pensamos. El receptor debe ser capaz de deducir el significado total de los textos que acompañan al resto de las medias.

- Economía de espera: evitar pausas e interrupciones bruscas, se debe respetar un ritmo rápido.

- Principio de múltiple entrada o multicanal: se debe utilizar diferentes canales para transmitir la misma información.

1.3 Metodologías de Desarrollo de Software

Según la Real Academia Española una metodología no es más que:

1. f. Conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal. [3]

Entiéndase, en este caso, por investigación científica el proceso de documentación y desarrollo de software.

"Las empresas que desarrollan software no pueden ignorar que su negocio es un negocio de software, y que el modelo que cada una adopte para las actividades de desarrollo y mantenimiento tiene implicaciones relevantes en la eficiencia general del negocio. El problema que pueden encontrar quienes deciden implantar métodos más eficientes, es caer en la desorientación ante el abanico de modelos de calidad, de procesos y de técnicas de trabajo desplegado en la última década; o abrazar al primero que se presenta en la puerta de la organización como "solución" de eficiencia y calidad". [4]

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Hoy en día existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo. Un ejemplo de ellas son las propuestas tradicionales centradas específicamente en el control del proceso. Estas han demostrado ser efectivas y necesarias en un gran número de proyectos, sobre todo aquellos proyectos de gran tamaño (respecto a tiempo y recursos). Sin embargo la experiencia ha demostrado que las metodologías tradicionales no ofrecen una buena solución para proyectos donde el entorno es volátil y donde los requisitos no se conocen con exactitud, porque no están pensadas para trabajar con incertidumbre.

Como respuesta a este problema surgieron otras metodologías que tratan de adaptarse a la realidad del desarrollo de software: las metodologías ágiles, que cambian significativamente algunos de los énfasis de los métodos ingenieriles. La diferencia inmediata es que son menos orientados al documento, exigiendo una cantidad más pequeña de documentación para una tarea dada. De muchas maneras son más bien

orientados al código: siguiendo un camino que dice que la parte importante de la documentación es el código fuente.

Aunque no se realizará una explicación detallada de todas las metodologías existentes se hace referencia a la Metodología OORAM (Object Oriented Role Analysis and Modeling), definida por Trygve Reenskaug, la cual lleva como principios que ningún objeto es una isla: el interés del objeto viene dado no por su estructura, sino por su forma de interactuar con el resto del sistema; divide y vencerás. Dicha metodología se encuentra dentro de la familia de metodologías orientadas al rol.

1.3.1 Proceso Unificado del Rational (RUP)

"RUP es un proceso de desarrollo de software…., es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos." [5]

Como todas las metodologías, RUP define claramente Quién (Trabajadores) debe hacer Qué (Artefactos), Cuándo (Flujos de trabajo) y Cómo (Actividades) debe hacerlo. Entre sus características fundamentales tiene que define un proceso centrado en la arquitectura (que relaciona la toma de decisiones que indican cómo tiene que ser construido el sistema y en qué orden) e iterativo e incremental, divide el proyecto en miniproyectos para cumplir así sus objetivos de manera más depurada.

El ciclo de vida de RUP

RUP divide el proceso en 4 fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor énfasis en los distintas actividades.

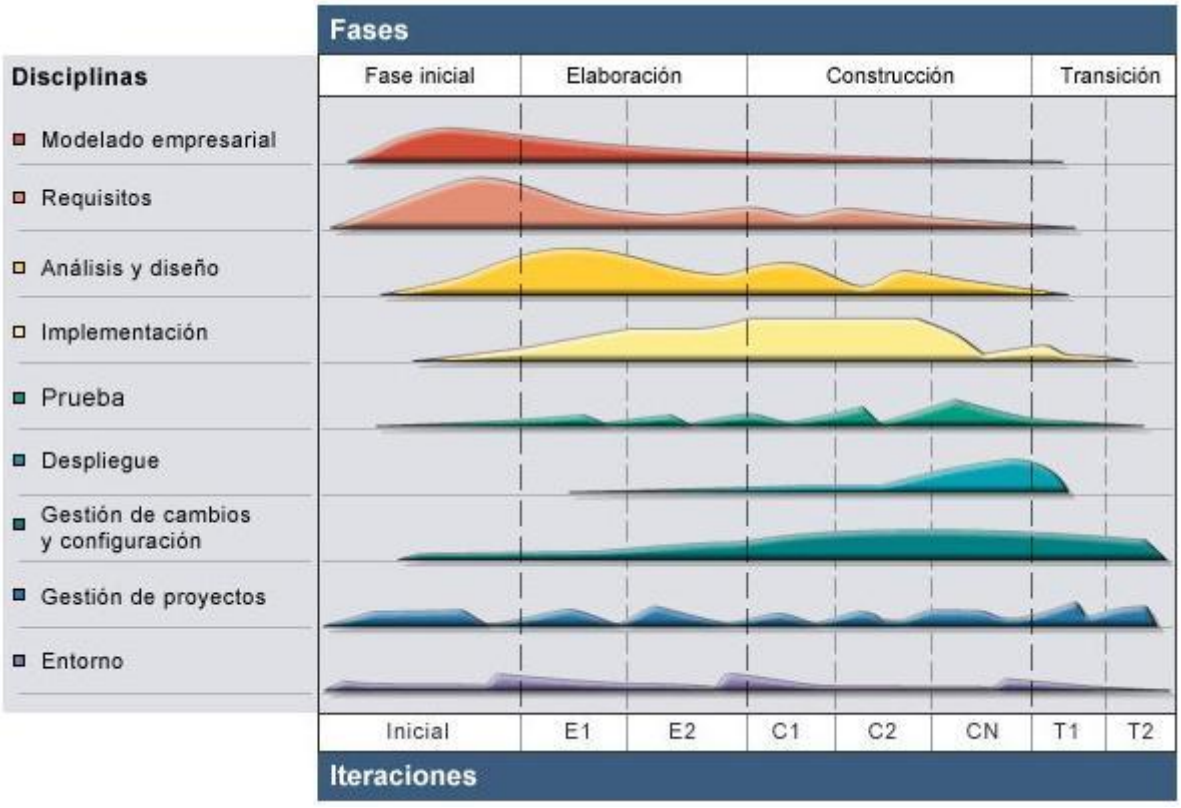


Figura 1: RUP en dos dimensiones. [6]

Inicio: Se hace un plan de fases, se identifican los principales casos de uso y se identifican los riesgos. Se define el alcance del proyecto.

Elaboración: Se hace un plan de proyecto, se completan los casos de uso y se eliminan los riesgos.

Construcción: Se concentra en la elaboración de un producto totalmente operativo y eficiente y el manual de usuario.

Transición: Se entrega el producto al cliente, se instala y se entrena a los usuarios. Como consecuencia de esto suelen surgir nuevos requisitos a ser analizados.

Dependiendo de la iteración del proceso, el equipo de desarrollo puede realizar 7 tipos de actividades en este: durante la fase de inicio las iteraciones están más enfocadas en las actividades de modelado del negocio o del dominio y de requisitos. En la fase de elaboración, las iteraciones se orientan al desarrollo de la *línea base* de la arquitectura, abarcan más los flujos de trabajo de Requerimientos, Modelamiento

del negocio (refinamiento), Análisis, Diseño y una parte de implementación orientado a la *línea base* de la arquitectura. Se especifican los requerimientos y se describen como se van a implementar en el sistema, se transforman al diseño del mismo, el cual se adapta para que sea consistente con el entorno de implementación. En la fase de construcción se implementan las clases definidas en el diseño. El resultado final es un sistema ejecutable. Posteriormente se inicia la fase de transición donde se realiza la entrega e instalación al cliente del sistema. El flujo de trabajo de Pruebas se ejecuta durante todas las fases de desarrollo, pues no es recomendable evaluar la calidad del producto sólo al final del proceso.

Durante todo el ciclo se lleva a cabo la Gestión de proyecto, que asegura el cumplimiento de los objetivos y la gestión de riesgos y restricciones que pudieran impedir obtener un producto acorde a los requisitos de los clientes. La configuración y control de cambios permite mantener la integridad de todos los artefactos que se crean en el proceso, y la información de cómo han evolucionado. Otro de los flujos de trabajo es Entorno, cuya finalidad es dar soporte al proyecto con las adecuadas herramientas, procesos y métodos. Brinda una especificación de las herramientas que se van a necesitar en cada momento. Estos son los llamados flujos de soporte ya que son el complemento de los flujos fundamentales de trabajo que establece RUP.

1.3.2 XP (Programación Extrema)

XP surge hace aproximadamente doce años y según palabras de su propio creador Kent Beck "*es una metodología ligera, eficiente, con bajo riesgo, flexible, predecible y divertida para desarrollar software*". [7]

Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural e inevitable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Las bases de esta metodología son sencillas, para muchos no es más que aplicar la lógica, reciclado continuo de código, comunicación y simplicidad, buscando en definitiva la reducción de costos.

Objetivos de XP:

XP cuenta con dos objetivos principales:

1. La satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.
2. Potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software. [8].

Fases de desarrollo de XP

Fase de Planificación

Historias de usuario: El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia.

Fase de Diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para obtener un diseño de fácil entendimiento e implementación que a la larga costará menos tiempo y esfuerzo desarrollar. También usar glosarios de términos y una correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

Fase de Codificación

El cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de XP. A la hora de codificar una historia de usuario su presencia es aún más necesaria, los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del

desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen las pruebas que verifiquen que la historia implementada cumple la funcionalidad especificada. La codificación debe hacerse atendiendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad.

Fase de Pruebas

Uno de los pilares de la metodología XP es el uso de pruebas para comprobar el funcionamiento de los códigos que se implementen. Se deben crear las aplicaciones que realizarán las pruebas con un entorno de desarrollo específico para las distintas clases del sistema omitiendo los métodos más triviales.

1.3.3 Metodología seleccionada

Luego de realizar el estudio de las metodologías antes mencionadas resultó ser RUP la más apropiada para el desarrollo del proceso actual; dado que desde muy temprano establece una sólida arquitectura, que al producirse cambios posteriores, no sufre un gran impacto. Otra de las características que se tuvo en cuenta para la selección de la metodología, es su enfoque iterativo e incremental, lo que permite trabajar en iteraciones de corta duración y que presentan metas bien definidas. En cada iteración existen entregables que deben ser legibles. Las iteraciones permiten detectar y corregir errores en etapas tempranas del desarrollo posibilitando no tener que realizar grandes cambios en fases avanzadas, una vez se ha concluido una iteración si estamos satisfechos iniciamos la siguiente hasta que se logra un producto de calidad.

La metodología XP a pesar de las ventajas que pudiera ofrecer no se adecúa al proceso actual debido a que el cliente debe formar parte activa en el proceso de desarrollo, requisito que no puede ser cumplido dado que el mismo se encuentra alejado del centro donde se desarrolla el proceso y se hace difícil la transportación. Por último mencionar la poca experiencia del equipo de desarrollo con esta metodología, que al utilizarla solo se lograría el atraso en la entrega del producto y su mala calidad. OORAM, al igual que XP, no es seleccionada debido a que propone un lenguaje de modelado, que además de ser privativo, es similar a UML; más adelante se especifica por qué UML no es el lenguaje seleccionado.

1.4 Lenguajes de Modelado de Software

A partir de la metodología seleccionada, RUP, es necesario un lenguaje para realizar el modelado del sistema que se va a implementar.

Previo a detenernos en el análisis de los diferentes lenguajes de modelado se hace necesario decir que un modelo es una abstracción del sistema que permite especificarlo a cierto nivel y que ayuda al ingeniero a “visualizar” el sistema a construir y si posee un gran nivel de abstracción puede ser utilizado también en la comunicación con el cliente.

Los modelos se usan para muchos propósitos:

- Para captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos.
- Para pensar en el diseño de un sistema.
- Para capturar decisiones del diseño en una forma mutable a partir de los requisitos.
- Para generar productos aprovechables para el trabajo.
- Para organizar, encontrar, filtrar, recuperar, examinar, y corregir la información en grandes sistemas.
- Para explorar económicamente múltiples soluciones.
- Para domesticar los sistemas complejos. [9]

1.4.1 Lenguaje Unificado de Modelado (UML: Unified Modeling Language)

“El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios.” [9] Este lenguaje se ha convertido en la notación estándar para definir, organizar y visualizar los elementos que configuran la arquitectura de un sistema. Un sistema es algo "compuesto", una construcción realizada por manos y herramientas siguiendo las directrices de un propósito. La palabra se aplica casi exclusivamente a

abstracciones con el fin de captar la totalidad de una realidad. A través de la notación UML podemos comunicar y compartir el conocimiento de una arquitectura gracias a la combinación simultánea de cinco vistas: *Vista de Casos de Uso*, *Vista de Implementación*, *Vista de Despliegue*, *Vista de Diseño*, y *Vista de Procesos* que están dirigidos a la modelación de los productos, a través de un conjunto de diagramas distribuidos por cada una de estas vistas.

En los años recientes, varios lenguajes de modelación orientada a objetos han surgido de los cuales UML es el último y más aceptado por la comunidad desarrolladora de sistemas informáticos de todo tipo. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como RUP), pero no especifica en sí mismo qué metodología o proceso usar.

A pesar de ser reconocido y aceptado internacionalmente por la mayoría de las comunidades de desarrollo de software gracias a su fuerte fundamento en la programación orientada a objetos y sus ventajas en la modelación de sistemas en todas sus fases de desarrollo, no es el lenguaje de modelado más adecuado para una aplicación con tecnología multimedia ya que en UML no se logran modelar eficientemente los entornos y características de este tipo de aplicaciones.

1.4.2 Lenguaje para la modelación Orientada a Objetos de Aplicaciones Multimedia: OMMMA – L

En la modelación de aplicaciones multimedia es necesario integrar varios aspectos: la integración temporal y sincronización de los diversos tipos de media utilizados, y sus diferentes características de tiempo. Varios modelos han sido propuestos para modelar aplicaciones multimedia. Predominantemente se concentran en modelar las relaciones temporales y la sincronización de las presentaciones multimedia; otros elaboran modelos que toman en cuenta la interactividad; otros se concentran en la estructura lógica

y conceptos de navegación en la hipermedia; pero hoy los lenguajes de modelación de software están normalmente basados en el paradigma Orientado a Objetos.

Desafortunadamente UML no soporta todos los aspectos de las aplicaciones multimedia de una forma adecuada e intuitiva. Especialmente, las características del lenguaje para modelar los aspectos de la interfaz de usuario, no se aplican explícitamente en los entornos multimedia. Otros conceptos de UML no son lo formalmente aplicables a las aplicaciones multimedia y de ser utilizados tal y como han sido planteados complicarían la modelación de este tipo de aplicaciones. Por estas razones se hizo necesario el desarrollo de una extensión de UML para este tipo de aplicaciones, denominada Lenguaje Orientado a Objetos para la Modelación de Aplicaciones Multimedia (OMMMA-L), que facilita el modelado de un gran rango de aspectos de aplicaciones multimedia interactivas de una forma integrada y comprensiva.

El Lenguaje Orientado a Objetos para Modelar Aplicaciones Multimedia (OMMMA-L), está sustentado en cuatro vistas fundamentales, donde cada una se asocia a un tipo de diagrama en particular.

- **Vista Lógica**

Modelada a través del Diagrama de Clases de OMMMA-L, extendido del Diagrama de Clases de UML, utilizando las mismas notaciones, pero incorporando las clases correspondientes a las medias: media continua y media discreta, generalizadas en una clase medias. Divide en dos áreas dicho diagrama: una para la jerarquía de los tipos de media y otra para la modelación de la estructura lógica del dominio de la aplicación.

- **Vista de Presentación espacial**

Modelada a través de los Diagramas de Presentación de OMMMA-L, los cuales son de nueva aparición en la extensión de UML, dado que este último no contiene un diagrama apropiado para esta tarea. Estos diagramas tienen el propósito de declarar las interfaces de usuario con un conjunto de estructuras delimitadas en tamaño y área, dividiéndose en objetos de visualización (texto, gráfico, video, animación) e interacción (scrolls, barras de menú, botones, campos de entrada y salida, hipertextos con hipervínculos). Estos diagramas de presentación pueden ser divididos en capas virtuales de presentación donde en cada uno de ellas sólo se haga referencia a una clase específica de componentes (por ejemplo, una vista para los objetos de visualización y otra para los de interacción, u otro tipo de división para la representación de los intereses de los desarrolladores).

- **Vista de Comportamiento temporal predefinido**

Modelada por el Diagrama de Secuencia de OMMMA-L, extendido a partir del diagrama de secuencia de UML. Modela una secuencia de una presentación predefinida dentro de una escena, donde todos los objetos dentro de un diagrama se relacionan al mismo eje de tiempo. En este diagrama se hace un refinamiento del eje del tiempo con la introducción de marcas a través de diferentes tipos de intervalos; marcas de inicio y fin de ejecución que permite soportar su reusabilidad; marcas de activación y desactivación de demoras en objetos de tipo media, posibilitando la modelación de las tolerancias de la variación de las restricciones de sincronización para los objetos media; activación compuesta de objetos media para la agrupación de objetos concurrentemente activos.

- **Vista de Control Interactivo**

Modelado a través del Diagrama de Estado, extendido a partir del diagrama de estado de UML, sintácticamente igual a este último, más con la diferencia semántica de que en el orden de unir los controles interactivos y predefinidos, no interrumpidos de los objetos, las acciones internas de estados simples tienen que llevar nombres de diagrama de secuencia en vez de diagramas de estado empotrados; queriendo esto decir que el comportamiento especificado por el diagrama de secuencia se provoca automáticamente cuando se entra al estado correspondiente donde se hace referencia. [10]

Las características de OMMMA-L pueden resumirse en:

- Soporta el modelado de los aspectos estructurales, funcionales y dinámicos de un sistema interactivo y su interfaz de usuario.
- Se concentra en la funcionalidad desde la perspectiva del sistema de software.
- Su sintaxis es definida explícitamente.
- Tiene una semántica informal e intuitiva.

1.4.3 ApEM-L: Lenguaje para la Modelación de Aplicaciones Multimedia Educativas

ApEM – L se presenta como una extensión de UML, tomando como bases teóricas principales OMMMA – L (2001) y OCL – 2.0 (2003), lo que produce las siguientes ventajas:

- Puede utilizar para su representación todas las herramientas CASE que existen actualmente para la modelación de UML.

- Es un lenguaje que utiliza el estándar internacional OCL, para la modelación de la programación Orientada a Objetos.
- No modifica la semántica del lenguaje base UML, sino que trabaja en estereotipos restrictivos, por lo que a su vez produce modificaciones descriptivas y decorativas en la representación de los componentes del lenguaje base. [11]

ApEM – L surge con el objetivo principal de desarrollar una extensión del lenguaje de modelado UML tomándolo como base e incorporando a este, a través de sus mecanismos de extensión, los elementos fundamentales del proceso productivo UCI. De esta forma se produjo un lenguaje de propósito particular para la modelación de aplicaciones con tecnología multimedia, fundamentalmente las educativas. Además de incorporar los elementos más significativos de extensiones anteriores como OMMMA – L y a su vez respetar lo establecido por el estándar OCL (2003) para de esta forma lograr una extensión consistente y escalable en el tiempo. Este lenguaje pretende convertirse en el área de la representación y la documentación de las aplicaciones con tecnología multimedia y no en un método de desarrollo de estas aplicaciones. Otro de los objetivos de ApEM – L es no suscribirse a un proceso de desarrollo en específico, sino expresarlo de manera tal que pueda ser utilizado con cualquiera de los existentes, aunque sugiere la utilización de procesos iterativos, incrementales y basados en prototipos que permitan la modelación de sistemas orientados a objetos.

Los conceptos y modelos de ApEM – L pueden agruparse en las siguientes áreas conceptuales:

- ***Estructura lógica*:** está compuesta por la vista estática y la vista de arquitectura. Cualquiera de los modelos presentados por ApEM – L define los conceptos claves de la aplicación que modela, las propiedades internas de estos y sus relaciones. Estos conceptos son modelados como clases, describiendo cada una un conjunto de objetos que almacenan información y se comunican para implementar su comportamiento
- ***Comportamiento dinámico*:** el comportamiento de la aplicación está descrito por la vista de comportamiento, la cual está compuesta por los diagramas de actividad, de secuencia, de colaboración y de estados, donde solo ha sido modificado el segundo de los listados anteriormente; adicionando una variable de tiempo donde quiera que sea necesario su especificación para un mejor entendimiento.

- ***Gestión del modelo*:** esta área es la que ha sufrido grandes cambios tanto en su carácter semántico como sintáctico, con la incorporación de estereotipos restrictivos en todos los diagramas a partir de nuevos conceptos incorporados a los diagramas de clases originales de UML. Se crean dos nuevos diagramas: Diagrama de estructura de presentación (DEP) y el Diagrama de estructura de navegación (DEN). [11]

Por conveniencia ApEM – L se ha dividido en varias vistas: estática, de arquitectura, de comportamiento y de presentación. Esta división se ha realizado en base a las áreas conceptuales anteriores.

- **Vista Estática:** está compuesta por el Diagrama de Clases, se divide en dos grandes zonas, la de la izquierda dedicada al árbol jerárquico de *las clases modelo entidad medias* que representan los recursos mediáticos de la aplicación y la zona de la derecha que se subdivide en cuatro zonas, la primera de estas dedicada a las clases *vistas*, a continuación y en el extremo superior derecho la zona dedicada a las clases *controladoras*, debajo la destinada a las clases *modelo*, quedando una banda inferior derecha dedicada en su extremo derecho a las clases *modelo entidad persistentes* para el tratamiento de la información persistente de la aplicación; y en el extremo izquierdo las clases correspondientes al Lenguaje de Alto Nivel (HLL en sus siglas en inglés correspondientes a High Level Language) con el que se programe (Lingo, ActionScript, C#, C++, Object Pascal, PHP, etc.).
- **Vista de Arquitectura:** está compuesta por el Diagrama de Componentes y el Diagrama de Despliegue, no sufriendo cambios este último. En el Diagrama de Componentes se incorporan restricciones en los tipos de componentes. Al seguir la arquitectura propuesta por el patrón MVC-E, normalmente los componentes podrán ser organizados por paquetes, que identificarían unidades físicas de encapsulamiento del código. Solo sería necesario luego un diagrama de componentes a un nivel de abstracción superior que represente como se comunican los distintos tipos de paquetes componentes del sistema.
- **Vista de Comportamiento:** está compuesta por cuatro diagramas: *de actividades*, *de estado*, *de secuencia* y *de colaboración*; siendo estos dos últimos generalizados en diagramas de interacción. En ApEM– L sólo ha sido modificado el diagrama de interacción de secuencia, con un estereotipo descriptivo y por ende decorativo para denotar el tiempo como variable de sumo interés en aplicaciones de este tipo.

- **Vista de Presentación:** ha sido incorporada completamente al lenguaje base UML, para permitir utilizar la semántica original de dicho lenguaje en la construcción de estructuras lógicas de presentación y navegación, incorporando un conjunto de estereotipos restrictivos y descriptivos para una mejor modelación, construyendo el *diagrama de estructura de navegación* y el *diagrama de estructura de presentación*. Con la incorporación de esta nueva vista queda resuelto el mayor problema de aplicaciones con estas características, y es la representación de los elementos de presentación y navegación, los cuales son de sumo interés para los desarrolladores. [11]

1.4.4 Selección del lenguaje de modelado

Se decide emplear ApEM– L como lenguaje para la modelación de la solución del problema en cuestión, ya que es un lenguaje extensivo de UML por lo que mantiene sus fundamentos y su estilo de trabajo, además tiene como bases teóricas OMMMA – L (2001) y OCL 2.0 (2003), lo que permite modelar aplicaciones con tecnologías multimedia que utilicen el paradigma orientado a objetos. Por otra parte puede utilizar para su representación todas las herramientas CASE que actualmente existen para la modelación de UML, no siendo así con OMMMA – L, ya que los diagramas de Presentación que define este lenguaje no pueden ser modelados con las herramientas existentes hoy. Al seleccionar este lenguaje se tuvo en cuenta también que el mismo trata de forma más detallada los conceptos referentes a software con tecnología multimedia, ofrece una visión mejor de los componentes al equipo de desarrollo en general y a los clientes. Además al definir nuevas clases y redefinir el patrón MVC_{MM} para representar exclusivamente los elementos de un software con tecnología multimedia, con características descriptivas, se logra un producto con alta calidad. Resuelve el principal problema en la modelación de este tipo de aplicaciones: los elementos de navegación y presentación, mediante la incorporación de los diagramas de Estructura de Navegación y de Estructura de Presentación. En las aplicaciones con tecnología multimedia lo más importante es el estímulo visual, proporcionado en este lenguaje por la Vista de Presentación, que además tiene asociada una descripción textual donde se incluye las descripciones de las medias y los elementos pedagógicos que se deben tener en cuenta en caso de ser un software educativo. Estas descripciones son similares al guión de contenido que se utiliza actualmente en aplicaciones de este tipo. Permite que el equipo de desarrollo pueda entender mejor la estructura y funcionamiento de la aplicación, dejando a un lado los casos de uso, que para aplicaciones con estas características no son lo más recomendado.

1.5 Tecnologías actuales para el desarrollo

Las aplicaciones multimedia han propiciado que las tecnologías de la información y las comunicaciones se desarrollen con una velocidad vertiginosa en los últimos tiempos; han logrado que los usuarios se interesen más por la investigación, sabiendo que pueden encontrar la información de diversas formas y no solo aquel texto estático, ahora unido al texto también aparecen imágenes, videos, audio que permiten obtener el conocimiento de forma amena y sencilla.

1.5.1 Herramientas de Autor

"En el campo informático se entiende como Herramienta de Autor, a todo Software que permite crear aplicaciones independientes del Software que lo generó" [12], es decir a la hora de ser ejecutadas no necesitan para nada de la herramienta creadora.

Los sistemas de autor utilizan estructuras para la creación de tareas comunes de las aplicaciones con tecnología multimedia.

- Carga y visualización de imágenes.
- Uso de efectos de transición en la navegación o presentación de la información.
- Ejecución y sincronización de archivos de sonido, video y animaciones.
- Definición de objetos visuales con funcionalidades específicas como son controles, objetos gráficos, contenedores y visualizadores de diferentes tipos de medias, ventanas y cuadros de diálogo.
- Recepción de entradas del usuario y asignación de acciones en calidad de respuesta por parte del sistema.

La realización de algunas de estas tareas en un lenguaje de programación sería muy complicada y llevaría muchas líneas de código pero en una herramienta de autor todo se simplificaría a unas pocas instrucciones y comandos.

1.5.1.1 Macromedia Flash 8.0

Macromedia Flash es una aplicación en forma de estudio de animación que trabaja sobre *"Fotogramas"* destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma…., utiliza gráficos vectoriales e imágenes ráster, sonido, código de

programa, flujo de vídeo y audio bidireccional. En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash. [13]

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia. Son también ampliamente utilizados en anuncios de la web.

Macromedia Flash se puede considerar por sus características como una herramienta de desarrollo completa, capaz de desarrollar aplicaciones con fuerte contenido interactivo y visual, y no solo animaciones para web de carácter publicitario y comercial.

Características principales:

- Interfaz gráfica amigable, potente y sencilla de usar.
- Soporta video con nuevas funcionalidades.
- Carga dinámica de imágenes, video y sonido.
- Previsualización de animaciones.
- Puede interactuar con una base de datos.
- Librería de símbolos.
- Soporte de audio MP3.
- Interacción con otros lenguajes como XML.

Flash 8 ofrece algunas ventajas:

- Diseños más atractivos.
- Optimización de fuentes.
- Bibliotecas integradas.
- Mayor potencia de animación.
- Mayor potencia gráfica.

- Mejoras en la importación de video.
- Compatibilidad Metadatos.
- Emulador para dispositivos móviles.
- Asistente de ActionScript. [14]

Con Macromedia Flash 8 se pueden realizar pruebas eficientes a las películas destinadas a dispositivos móviles como los celulares, gracias a Flash Lite, nuevo emulador que es incorporado a esta herramienta, lo que permite hacer publicaciones que ya no contengan errores. Por otra parte gracias a su compatibilidad metadatos se pueden incluir los archivos SWF en internet con un título, descripción y/o palabras clave para que los motores de búsqueda reflejen con más precisión el contenido representado por el archivo. [14]

1.5.1.2 Macromedia Director

Macromedia Director es una potente herramienta con la se puede hacer y distribuir presentaciones multimedia únicas, tanto en CD-ROM, DVD, como a través del Web. [15]

También se pueden combinar imágenes, sonidos, animaciones, textos y videos, todo en un único archivo, y luego exportarlo en varios tipos de ficheros, incluyendo AVI.

Alguna de las cualidades que podemos destacar de Macromedia Director, es la gran calidad de reproducción de las presentaciones gracias al potente motor que incorpora la aplicación. Además, posee efectos visuales de muy buena calidad, soporte para ilustraciones vectoriales etc. Director permite personalizar el tiempo que se tome la aplicación para añadir soporte a los elementos interactivos más importantes, como la navegación, controles personalizables y listados.

Además del potente lenguaje incorporado (Lingo), una de sus principales ventajas radica en el uso de los llamados xtras. Se trata de “pequeños programas” desarrollados en lenguaje C++ por otros usuarios o terceras empresas, que proporcionan al usuario infinidad de utilidades. [16]

A pesar de las grandes posibilidades y recursos de Director para el desarrollo de aplicaciones con tecnología multimedia, tienen algunas desventajas como la falta de un mejor editor de imágenes y su dificultad de uso para usuarios poco experimentados, y su principal desventaja es que no es multiplataforma.

1.5.1.3 Mediator 6 EX

El usuario tiene las herramientas necesarias para crear presentaciones con efectos especiales, además, de comenzar el trabajo con variables y el uso de los Scripts, tiene la capacidad para programar y escribir códigos y así lograr que las presentaciones tengan un aspecto aun más refinado. Aunque una de las principales desventajas que presenta esta herramienta es que las aplicaciones generadas no son multiplataforma.

Mediator 6 EX, creado por MatchWare en su versión 1993-2001. Este programa consta a su vez de dos programas: Diseñador de Mediator (Mediator Designer) y el espectador de Mediator (Mediator Viewer) [17] que permiten crear y ver el producto final, respectivamente.

El Diseñador de Mediator (Mediator Designer) es donde se crean los proyectos. Este modo también incluye el modo de prueba, que es donde se prueba el proyecto que va a ser diseñado, este puede compararse con el espectador, solo que el propósito es ir probando el proyecto dentro del diseñador, sin necesidad de buscar el archivo para ejecutarlo. El espectador de Mediator (Mediator Viewer) es donde se muestra el proyecto después de haber guardado el archivo. [17]

Trae incorporado funcionalidades como:

- Incorporar objetos a las páginas.
- Ordenamiento de las páginas
- Trae eventos definidos
- Variables de sistema.
- Uso de la herramienta List Box y uso de recursos predefinidos.
- Acceso a bases de datos
- Creación de scripts.

1.5.2 Selección de la herramienta para el desarrollo de la aplicación

Se selecciona como herramienta de autor para el desarrollo de la aplicación Macromedia Flash 8.0, debido a que soporta flujo progresivo por defecto, lo cual define que los fotogramas son cargados individualmente y pueden ser mostrados antes de que se cargue todo el archivo, el empleo de gráficos

vectoriales hace que los archivos ocupen menos espacio y consuman menos ancho de banda al ser trasmitidos, soporta funcionalidades para la carga de datos a través de XML, sonido MP3, imágenes de múltiple formato, así como otras aplicaciones generadas en Flash. Otra de las grandes ventajas de Flash 8 es que es multiplataforma y posee un entorno de fácil uso. El aumento de hasta 10 veces más en la velocidad de las películas también es una ventaja que proporciona Flash 8. Por último destacar que en esta herramienta es donde el equipo de desarrollo tiene más experiencia.

1.6 Lenguajes de Programación

1.6.1 Lenguaje XML (Lenguaje de Marcas Extensible)

XML, sigla en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. [18]

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo.

Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen más grande y con unas posibilidades mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Ventajas de XML:

- Es extensible: después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML, lo que posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones.

- No requiere licencias, es independiente de la plataforma y tiene un amplio soporte.
- Debido a su alta portabilidad de datos es considerado como una base de datos.

XML, como lenguaje de marcas estándar para el intercambio de información entre aplicaciones, no es una excepción al soporte de ActionScript. El uso del objeto XML, destinado exclusivamente a la gestión de archivos y contenidos formateados en este estándar, permite a una película Flash importar y exportar fácilmente información desde y hacia lenguajes de servidor o bases de datos. XML se encarga de estructurar estos datos de forma tal que puedan ser leídos e interpretados sin problemas por cada una de las partes.

Lectores dinámicos de noticias, sistemas de gestión de weblogs y foros son algunas de las aplicaciones donde el uso del lenguaje XML se hace prácticamente imprescindible. No obstante, su campo de acción no se limita únicamente a las aplicaciones de carácter dinámico. Una de las principales razones por la que se recomienda la integración de XML con ActionScript es evitar una recurrente edición del archivo fuente (.fla) cada vez que se necesite introducir algún cambio en el contenido.

1.6.2 ActionScript 2.0

ActionScript es el lenguaje de programación utilizado para desarrollar aplicaciones web animadas realizadas en el entorno de Flash. Fue introducido por Macromedia Flash 4, siendo desde la versión de Flash MX 2004 un verdadero lenguaje de programación orientada a objetos, y desde entonces hasta ahora, ha ido ampliándose poco a poco, hasta llegar a niveles de dinamismo y versatilidad muy altos. [19]

Como lenguaje de script no requiere la creación de un programa completo para que la aplicación alcance los objetivos. El lenguaje está basado en especificaciones de estándar de industria ECMA-262, un estándar para JavaScript.

Flash está compuesto por objetos, con su respectiva ruta dentro del swf. Cada uno de estos objetos en ActionScript pertenece a una clase (MovieClip, Botones, Vectores (Arrays), etc.), que contiene Propiedades y Métodos.

- Propiedades: dentro del archivo raíz de la clase, están declaradas como variables (_alpha, useHandCursor, length).

- Métodos: Dentro del archivo raíz de la clase, están declaradas como funciones (stop (), gotoAndPlay (), getURL ()).

ActionScript 2.0 es una gran actualización al lenguaje de scripts o secuencias de comandos de Flash, que mejora de forma radical el desarrollo orientado a objetos mediante la formalización de la sintaxis y la metodología de la Programación Orientada a Objetos (OOP). Con una orientación eminentemente práctica, ActionScript 2.0 resulta además el aliado perfecto para ajustar y crear la arquitectura de un proyecto orientado a objetos, así como para comprender cómo los componentes de interfaz y las subclases de clip de película se relacionan en una aplicación de Flash bien estructurada.

1.7 Herramienta de modelado de software

1.7.1 Rational Rose

Rational Rose forma parte de las herramientas CASE que soportan la especificación completa de UML, ventaja que permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común.

Es la mejor elección para el ambiente de modelado que soporte la generación de código a partir de modelos en Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Rational Rose proporciona un lenguaje común de modelado para el equipo que facilita la creación de software con calidad y rapidez. Permite hacer un seguimiento completo del tiempo de desarrollo, permite entender la estructura de sistemas, y ayuda al equipo de desarrollo a mantener una estrecha relación durante todas sus fases así como mantener la consistencia de los modelos del sistema, revisión de la sintaxis UML y generar la documentación automáticamente.

Otra ventaja de Rose es la buena integración de modelos con otros componentes del proyecto luego de la modelación de estos y sus interfaces de forma individual.

1.8 Conclusiones del Capítulo

Se abordaron los elementos teóricos a partir de los cuales se fundamentará el desarrollo del proceso de software, quedando definida RUP como metodología de desarrollo. Se utilizará como lenguaje de modelado ApEM-L, y Rational Rose será la herramienta a emplear para la modelación e integración del sistema. La herramienta de autor seleccionada para la creación de la aplicación, es Macromedia Flash

8.0, conjuntamente con ActionScript 2.0 como lenguaje de programación y XML para el tratamiento e intercambio de datos.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

Describir la solución propuesta para resolver el problema científico planteado es el objetivo del presente capítulo, dicha solución está guiada por el Proceso Unificado.

Algunas de las temáticas a tratar en el capítulo serán la identificación de los requisitos funcionales y los requisitos no funcionales del sistema, así como conceptos asociados al dominio. Además se pretende modelar el dominio a partir de los conceptos identificados, realizar los diagramas de estructura de navegación, representar la Vista de Gestión del Modelo, en general, describir el sistema propuesto.

2.2 Solución propuesta

Para resolver el problema de que no exista un producto informático que permita a especialistas e interesados estudiar la Colección de Corales Pétreos existente en nuestro país, y que además permita una mejor conservación de la misma, se propone la realización del software con tecnología multimedia “Colección de Corales Pétreos”.

El producto está dirigido, como ya se especificó, a especialistas y a todos aquellos que de una u otra forma estén interesados en el tema y que tengan un conocimiento básico sobre computación para navegar en el software. Permitirá acceder al contenido de cada especie, entiéndase por la información asociada a cada una de ellas y las imágenes en su medio natural y en su lugar de conservación; y conocer su sistemática, realizar búsquedas de palabras o frases dentro de cada clase de corales, así como imprimir el contenido de una especie. También permitirá consultar un glosario de términos específicos de la colección así como conocer sobre la historia de esta. Para hacer el producto más ameno, brindará la opción de escuchar un tema musical de fondo al iniciar la aplicación, teniendo la posibilidad de desactivarlo o activarlo según se desee.

A continuación se muestra una imagen de la aplicación.

Figura 2: Imagen de la aplicación.

2.3 Descripción de un modelo de dominio

Todo sistema puede llegar a ser complejo, por lo que si se pretende gestionar su complejidad y comprenderlo se necesita dividirlo en pequeñas partes, que pueden ser representadas a través de modelos.

Un modelo no es más que “una abstracción del sistema, especificando el sistema modelado desde un cierto punto de vista y en un determinado nivel de abstracción” [20] y sirve para captar y enumerar exhaustivamente los requisitos y el dominio de conocimiento, de forma que todos los implicados puedan entenderlos y estar de acuerdo con ellos [9]. RUP, como metodología de desarrollo propone, en caso de

que los procesos dentro del sistema estén bien definidos, la realización de un modelo de negocio, y en el caso de que los procesos no puedan ser claramente identificados un modelo de dominio.

Para el sistema con tecnología multimedia propuesto sólo se identificaron conceptos y objetos relacionados con el tema, no logrando definir procesos específicos del sistema, por lo que se propone la realización de un modelo de dominio.

Un modelo de dominio, o modelo conceptual, sólo tiene en cuenta los tipos de objetos más importantes del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en el que trabaja el sistema. Dicho modelo representa conceptos del mundo real y se describe mediante diagramas UML, en especial diagramas de clases. [5]

2.3.1 Identificación de los conceptos del dominio

Clase: Categoría taxonómica superior que reúne varios Ordenes.

Orden: Categoría taxonómica superior que contiene dentro de sí varias Familias con características similares.

Familia: Se define como una categoría sistemática que incluye un género o un grupo de ellos con un origen filogenético común y que está separado de otras familias por un espacio o vacío determinado.

Género: Es una unidad taxonómica colectiva constituida por un número de especies similares y relacionadas. Visto por los evolucionistas es un grupo de especies que descienden de un ancestro común.

Especie: Es la categoría taxonómica más importante en la biología. Se define como grupos de poblaciones naturales de actual o potencial entrecruzamiento que se reproducen aisladas de otros grupos similares.

Subespecie: Son agrupaciones de poblaciones locales geográficamente definidas que difieren taxonómicamente en algunas características la especie.

Forma: Se considera un “término neutral”, es decir que se usan en la taxonomía de manera informal, particularmente en aquellos casos en que faltan datos o pruebas para la clasificación como subespecie o especie. Se utiliza la forma para una unidad, o un grupo o un número de unidades.

Colección de corales: Conjunto de corales relacionados entre sí.

Especialistas e interesados: Son todas aquellas personas que de una u otra forma están interesadas, en este caso específico, en la colección.

Contenido: Información que contiene el producto multimedia relacionado con cada especie en particular.

Historia: Breve reseña de la colección desde su creación.

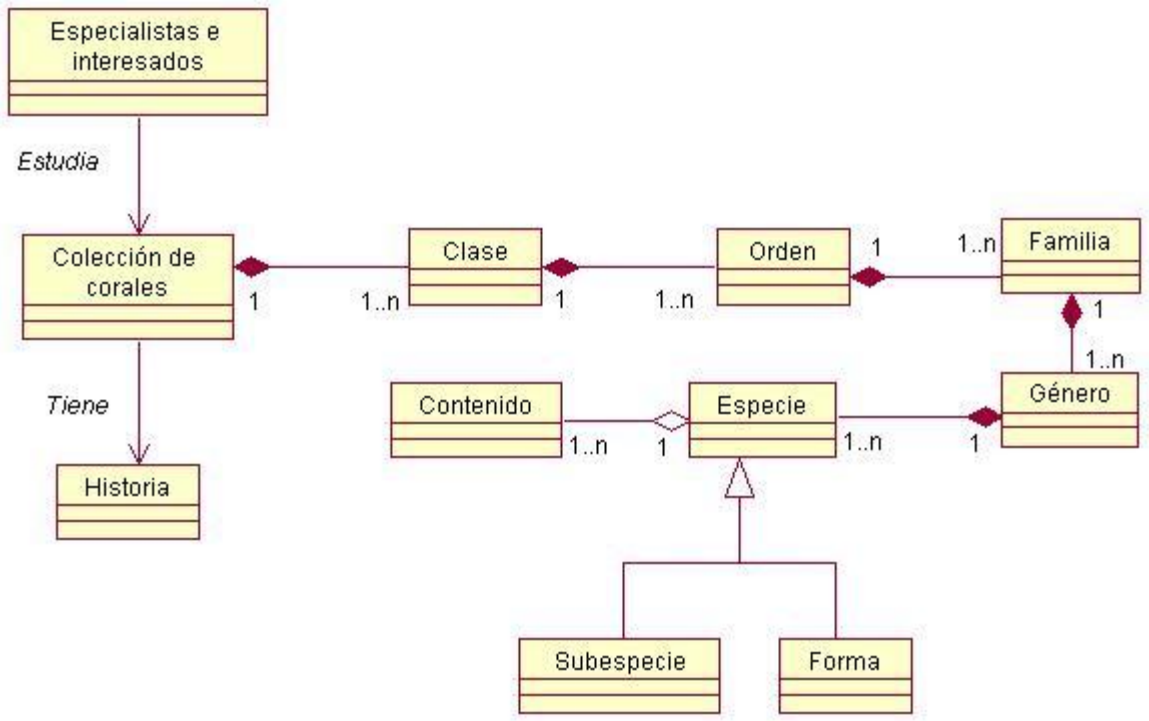


Figura 3: Modelo del dominio.

2.4 Requerimientos del sistema

Los requerimientos o requisitos, según la definición dada por la IEEE Standard Glossary of Software Engineering Terminology, son las condiciones o capacidades que debe cumplir un sistema para resolver un problema o lograr un objetivo especificado por el usuario.

La captura de requisitos “es el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir”. [5]

Un producto informático puede fracasar debido a varias razones, una de ellas, la mala planificación, pero una de las que más golpea a la industria del software actual es la deficiente identificación de los

requerimientos. Muchas veces lo que interpreta el equipo de desarrollo y el producto final que se obtiene no es, ni remotamente, lo que el cliente desea, lo que necesita. Al realizar una correcta identificación de los requerimientos se logra una mejor comunicación entre los miembros del equipo de desarrollo y de estos con el cliente.

Los requerimientos se clasifican en dos grupos, los requerimientos funcionales y los no funcionales.

Los requerimientos funcionales especifican acciones que el sistema debe ser capaz de cumplir, el comportamiento de entrada y salida del sistema, es decir, las acciones que debe realizar para que el usuario obtenga el resultado deseado.

Los requerimientos no funcionales son propiedades que el producto debe tener, no son más que características que hacen al producto usable, confiable, rápido y atractivo. Son los que permiten que el producto tenga buena aceptación o no.

2.4.1 Requerimientos Funcionales

R1: Mostrar presentación de la aplicación.

R2: Mostrar breve reseña histórica de la colección.

R3: Permitir la fácil navegación a través de cada una de las clases de corales pétreos hasta las especies.

R3.1: Mostrar la jerarquía existente desde las clases hasta las especies.

R4: Mostrar el contenido referente a cada una de las especies.

R4.1: Mostrar en la pantalla de inicio de cada clase el o los órdenes que lo componen.

R4.2: Mostrar las familias pertenecientes a cada orden.

R4.3: Mostrar los géneros asociados a cada familia.

R4.4: Mostrar el conjunto de especies del género seleccionado.

R4.5: Mostrar el texto del contenido de la especie seleccionada por el usuario.

R5: Permitir realizar búsquedas dentro de cada una de las clases.

R5.1: Realizar búsqueda por un criterio dado: palabra o frase.

R5.2: Mostrar los resultados de la búsqueda en una ventana emergente.

R5.3: Permitir el acceso, a través de la ventana emergente, al contenido de las especies según selección del usuario.

R6: Permitir al usuario el control del audio de fondo.

R6.1: Permitir al usuario poder activar o desactivar el audio.

R7: Permitir la opción de imprimir el contenido incluido en la aplicación.

R7.1: Permitir imprimir el contenido de una especie.

R8: Permitir al usuario abandonar la aplicación desde cualquier pantalla en que se encuentre.

R8.1: Mostrar la opción de confirmación de salida de la aplicación al usuario.

R9: Mostrar créditos una vez confirmada la salida de la aplicación.

R10: Mostrar imágenes de cada una de las especies que componen la colección.

R11: Mostrar el mapa de ubicación de las especies.

R12: Permitir restaurar o maximizar la aplicación.

R13: Mostrar glosario de términos.

2.4.2 Requerimientos No Funcionales

- Requerimientos de Portabilidad
  - El producto podrá ser usado bajo los sistemas operativos Windows y Linux.
- Requerimientos de hardware
  - Procesador Pentium II o superior.
  - Lector de CD-ROM o de DVD, en caso de portar el producto en un dispositivo de almacenamiento externo USB se requerirá de un puerto USB.
  - Mouse (Ratón) y teclado.
  - 64 MB de RAM o más.
- Requerimientos de software

  - Sistema Operativo Windows 98 o superior o cualquiera de las distribuciones de Linux a partir del año 2003.

- Restricciones en el diseño y la implementación
  - El producto se desarrollará con Macromedia Flash 8.0.
  - El lenguaje de programación será ActionScript 2.0.
  - Para la gestión de la información se utilizará el lenguaje XML 1.0.
- Requerimientos de usabilidad
  - Los usuarios del producto deberán tener un conocimiento básico del manejo de las computadoras y los sistemas operativos.
  - La aplicación deberá tener una interfaz y navegación asequibles, amenas y funcionales tanto para usuarios expertos como para novatos.
- Requerimientos de interfaz de hardware
  - El sistema requiere de una impresora para el caso en que se desee imprimir cualquier contenido de la aplicación.
- Requerimientos de interfaz externa
  - La aplicación contará con una interfaz sencilla y amigable.
  - El diseño permitirá la distinción visual de los elementos del sistema.
  - El sistema proporcionará la información de forma clara y legible, permitiendo a los usuarios una interpretación correcta de la misma.
- Requerimientos de rendimiento
  - El sistema deberá dar respuesta de forma ágil a las acciones indicadas por el usuario.

2.5 Vista de Gestión del Modelo

Dentro de los tantos aportes brindados por ApEM – L al Lenguaje Unificado de Modelado (UML) se encuentran las modificaciones realizadas a la Vista de Gestión del Modelo (VGM) tanto en su carácter semántico como sintáctico, con la incorporación de estereotipos restrictivos en todos los diagramas a

partir de nuevos conceptos incorporados a los diagramas de clases originales o básicos de UML. [11] La VGM representa en esencia la división de la aplicación en subsistemas y la relación entre ellos. En el caso específico del producto que se desarrolla como parte del presente trabajo investigativo se identificaron tres subsistemas, definidos por las características y las funcionalidades de las vistas que los conforman.

El subsistema Principal contiene la presentación, lo referente a la historia de la colección y las opciones principales como puede ser la música, el imprimir, el glosario de términos, volver al inicio, restaurar y cerrar. Este subsistema cuenta con las siguientes vistas de presentación: Vista de Presentación Presentación, Vista de Presentación Historia, Vista de Presentación Principal y la Vista de Presentación Salir. Por su parte el subsistema Contenido agrupa lo referente a la navegación entre la galería de imágenes, el mapa, el contenido de cada especie, además de incluir la opción de la búsqueda. Este contiene: la Vista de Presentación Mapa, Vista de Presentación Galería, Vista de Presentación Información, Vista de Presentación Común, Vista de Presentación ResultadosBúsqueda.

Figura 4: Vista de Gestión del Modelo

2.6 Diagramas de Estructura de Navegación (DEN)

“Los diagramas clásicos provistos por UML, como el diagrama de clases o el de estado no son suficientes para modelar aspectos de los sistemas multimedia, por ejemplo, para modelar el espacio de navegación y para representar este modelo gráficamente.” [21]

Para dar solución a este problema ApEM – L define el Diagrama de Estructura de Navegación el cual además de conservar las clases ya definidas, modelo – entidad – media texto y modelo – entidad – media imagen, identifica otras nuevas: clase menú, clase índice, clase consulta y clase botón las cuales permiten representar con mayor detalle la navegación en el producto.

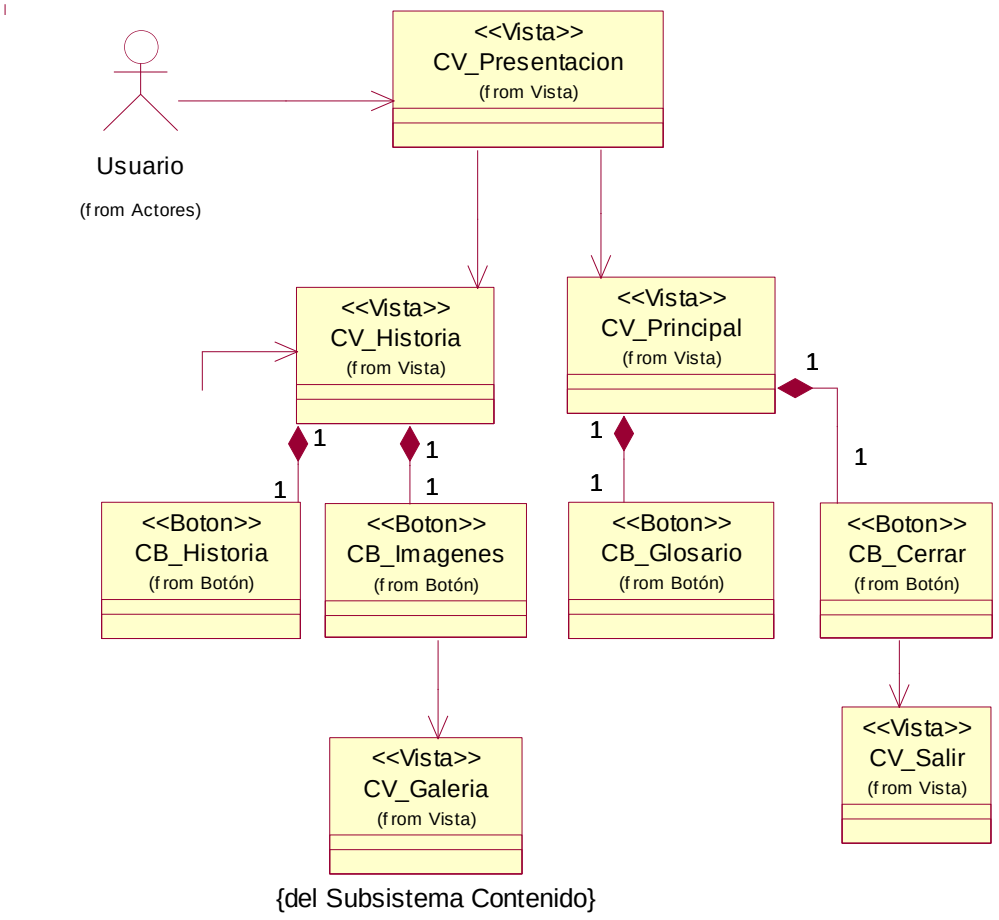


Figura 5: DEN Subsistema Principal

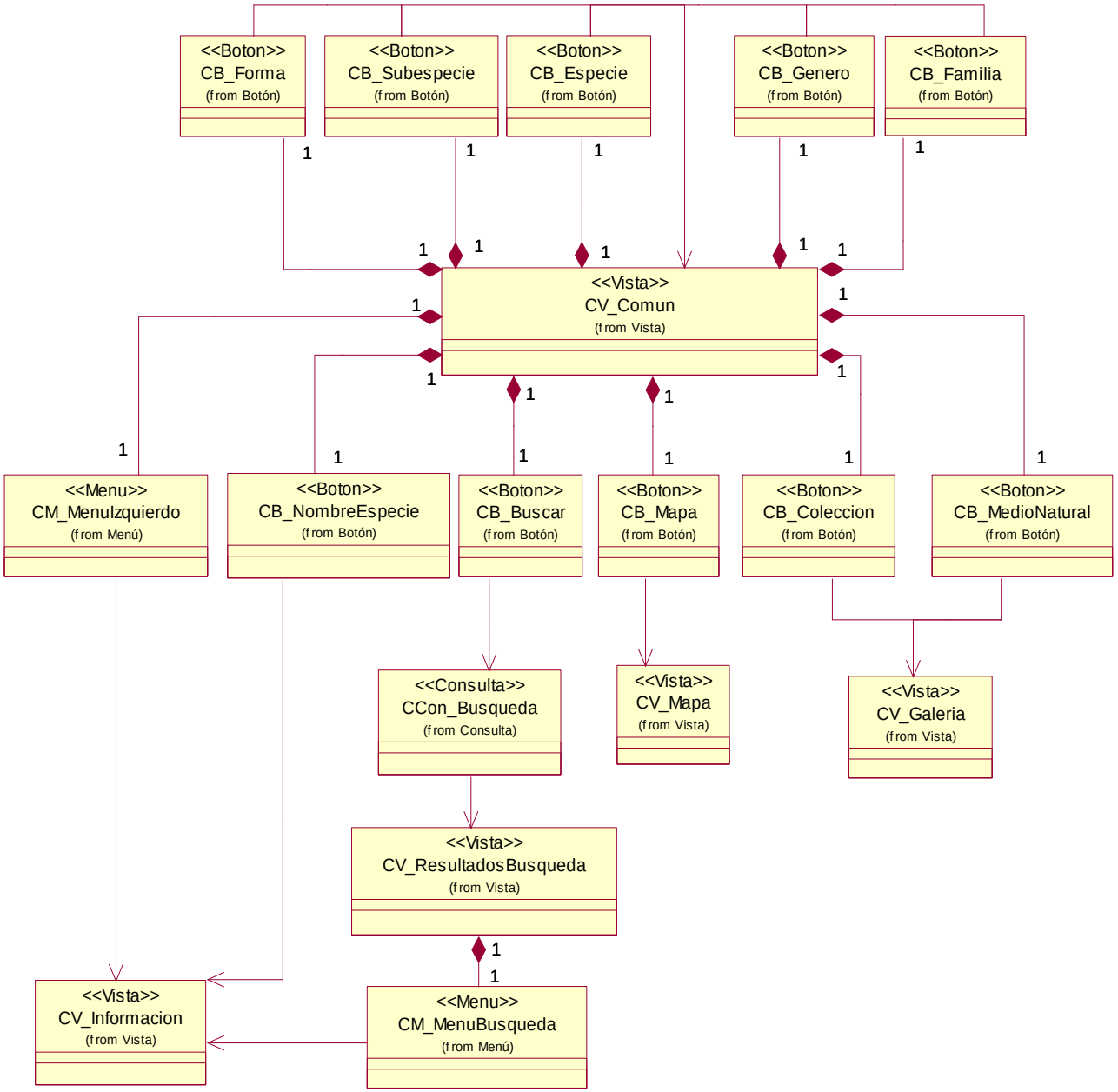


Figura 6: DEN Subsistema Contenido

2.7 Justificación de los actores del sistema

Actor: Usuario

Descripción: Representa a todas aquellas personas que interactúen con la aplicación y que de una u otra forma serán beneficiadas al obtener la información que se brinda.

2.8 Descripción textual de las Vistas de Presentación

Tabla 1: Descripción textual de la Vista de Presentación Presentación

Descripción textual de la Vista Presentación		
Actores de la Vista de Presentación	Usuario	
Propósito	Permitir al usuario observar la presentación de la aplicación y acceder a la pantalla inicial de la misma.	
Objetivos Pedagógicos		
Resumen	La vista se inicia cuando el usuario ejecuta la aplicación. El sistema muestra la animación de presentación y luego da paso a la Vista de Presentación Historia.	
Vistas asociadas	Vista de Presentación Historia, Vista de Presentación Principal.	
Referencias	R1	
Precondiciones		
Poscondiciones	Se ejecuta la presentación de la aplicación.	
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Ejecuta la aplicación.		
	2. Muestra la animación de presentación.	A_Presentación

	3. Inicia Vista de Presentación Historia y Vista de Presentación Principal.	
Curso alterno de los eventos		
Acción	**Curso alterno**	
Prioridad	Crítica	
Mejoras		
Medias a utilizar		
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación	A_Presentación	Animación que muestra los logotipos de las instituciones implicadas.	En construcción.
Sonido			
Texto			
Reglas Pedagógicas			

Tabla 2: Descripción textual de la Vista de Presentación Historia

Descripción textual de la Vista de Presentación Historia	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario observar una breve reseña histórica de la colección.
Objetivos Pedagógicos	
Resumen	La vista se inicia cuando culmina la Vista Presentación. Muestra información con una

	reseña histórica de la colección.	
Vistas asociadas	Vista de Presentación Principal, Vista Presentación.	
Referencias	R2	
Precondiciones	Debe haberse ejecutado la Vista Presentación.	
Poscondiciones	Se muestra la historia de la colección.	
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
	1. Muestra un área de texto con reseña histórica de la colección.	T_Historia
	2. Muestra la opción de ver imágenes históricas de la colección.	
3. Si selecciona la opción para ver las imágenes.		
	4. Muestra Vista de Presentación Galería (Subsistema Contenido).	
5. Si selecciona la opción de salir de la vista.		
	6. Termina la vista.	
Curso alterno de los eventos		
Acción	**Curso alterno**	
Prioridad	Crítica	

Mejoras			
Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación			
Sonido			
Texto	T_Historia	Texto que contiene una breve reseña histórica de la colección de corales.	Existe.
Reglas Pedagógicas			

Tabla 3: Descripción textual de la Vista de Presentación Principal

Descripción textual de la Vista de Presentación Principal		
Actores de la Vista de Presentación	Usuario	
Propósito	Permitir al usuario seleccionar opciones como cerrar, imprimir, ver glosario de términos, activar y desactivar sonido de fondo, maximizar o restaurar la aplicación y seleccionar alguna de las Clases de Corales para ver su información.	
Objetivos Pedagógicos		
Resumen	La vista se inicia al culminar la ejecución de la Vista Presentación.	
Vistas asociadas	Vista Presentación, Vista Historia, Subsistema Contenido.	
Referencias	R6, R7, R8, R12, R13	
Precondiciones	Debe haberse ejecutado la Vista Presentación.	
Poscondiciones		
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
	1. Muestra las diferentes opciones que puede ejecutar el usuario.	
2. Si selecciona la opción de salir de la aplicación.		
	3. Se inicia la Vista de Presentación Salir.	

4. Si selecciona la opción del sonido.		
	5. Se activa o desactiva el sonido dependiendo de cómo se encuentre este.	
6. Si selecciona la opción de imprimir.		
	7. Se imprime la información que se muestra en ese momento.	
8. Si selecciona la opción de restaurar o maximizar.		
	9. Se restaura o maximiza la aplicación.	
10. Si selecciona la opción del glosario.		
	11. Se muestra el glosario de términos.	
12. Si selecciona la opción de una de las Clases de corales.		
	13. Se muestra el contenido de la Clase seleccionada.	T_Clase
Curso alterno de los eventos		
Acción	**Curso alterno**	

Prioridad		Crítica	
Mejoras			
Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación			
Sonido			
Texto	T_Clase	Contiene información textual sobre una Clase	Existe
Reglas Pedagógicas			

Tabla 4: Descripción textual de la Vista de Presentación Mapa

Descripción textual de la Vista de Presentación Mapa	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario observar la ubicación geográfica de las especies en el mapa de Cuba.
Objetivos Pedagógicos	
Resumen	La vista se inicia cuando el usuario selecciona la opción mapa. El sistema muestra el mapa con la distribución de la especie.
Vistas asociadas	Vista Principal (Subsistema Principal), Vista Común.
Referencias	R11
Precondiciones	
Poscondiciones	Se muestra el mapa de Cuba con la distribución de la especie a lo largo del mismo.

Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Ejecuta la opción mapa		
	2. Muestra el mapa con la distribución de la especie	I_Mapa

Curso alterno de los eventos	
Acción	**Curso alterno**
Prioridad	Crítica
Mejoras	

Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen	I_Mapa	Mapa que muestra la distribución geográfica de la especie en la isla de Cuba.	Existe.
Video o Animación			
Sonido			
Texto			

Reglas Pedagógicas

Tabla 5: Descripción textual de la Vista de Presentación Galería

Descripción textual de la Vista de Presentación Galería	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario observar imágenes de la colección.
Objetivos Pedagógicos	

Resumen	La vista se inicia cuando el usuario selecciona la opción galería. El sistema muestra imágenes que representan a la especie en su medio natural y en la colección, así como imágenes históricas de la misma.	
Vistas asociadas	Vista Principal (Subsistema Principal), Vista Común.	
Referencias	R10	
Precondiciones		
Poscondiciones	Se muestran imágenes de la colección.	
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Ejecuta la opción galería.		
	2. Muestra las imágenes que representan a la especie en su medio natural y en la colección.	I_Imágenes
3. Si selecciona la opción imágenes de la Vista de Presentación Historia.		
	4. Muestra imágenes históricas de la colección.	I_Imágenes_historia
Curso alterno de los eventos		
Acción	**Curso alterno**	
Prioridad	Crítica	
Mejoras		

Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen	I_Imágenes	Imágenes de la especie en su medio natural y en la colección.	Existe.
Imagen	I_Imágenes_historia	Grupo de imágenes que muestran la historia de la colección.	Existe
Video o Animación			
Sonido			
Texto			
Reglas Pedagógicas			

Tabla 6: Descripción textual de la Vista de Presentación Información

Descripción textual de la Vista de Presentación Información	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario tener acceso a la información de cada una de las especies que conforman la colección.
Objetivos Pedagógicos	
Resumen	La vista se inicia cuando el usuario selecciona la opción para ver la información de una especie determinada. El sistema muestra información referente a la especie.
Vistas asociadas	Vista Principal (Subsistema Principal), Vista Común.
Referencias	R4
Precondiciones	

Poscondiciones	Se muestra información referente a una especie.

Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Ejecuta la opción para ver información de una especie.		
	2. Muestra la información asociada a la especie.	I_Imagen, T_Información

Curso alterno de los eventos	
Acción	**Curso alterno**
Prioridad	Crítica
Mejoras	

Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen	I_Imagen	Imagen representativa de la especie.	Existe.
Video o Animación			
Sonido			
Texto	T_Información	Texto correspondiente a la información de la especie.	Existe.

Reglas Pedagógicas

Tabla 7: Descripción textual de la Vista de Presentación Común

Descripción textual de la Vista de Presentación Común

Actores de la Vista de Presentación	Usuario	
Propósito	Permitir al usuario navegar entre la Vista de Presentación Información, la Vista de Presentación Galería, la Vista de Presentación Mapa y la Vista de Presentación ResultadosBúsqueda.	
Objetivos Pedagógicos		
Resumen	La vista se inicia cuando culmina la Vista de Presentación Historia.	
Vistas asociadas	Vista de Presentación Principal (Subsistema Contenido), Vista de Presentación Salir (Subsistema Contenido), Vista de Presentación Mapa, Vista de Presentación Galería, Vista de Presentación Información, Vista de Presentación ResultadosBúsqueda.	
Referencias	R3	
Precondiciones		
Poscondiciones	Se ejecuta una de las opciones de la vista.	
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Si selecciona la opción de ver el contenido de una familia.		
	2. Muestra los géneros dentro de una familia.	
3. Selecciona ver el contenido de un género.		
	4. Muestra las especies contenidas dentro del género.	

5. Selecciona la opción para ver el contenido de la especie.		
	6. Se inicia la Vista de Presentación Información.	
7. Si selecciona la opción de la galería.		
	8. Se inicia la Vista de Presentación Galería.	
9. Si selecciona la opción de ver el mapa.		
	10. Se inicia Vista de Presentación Mapa.	
11. Si selecciona la opción de realizar búsquedas.		
	12. Se inicia la Vista de Presentación ResultadosBúsqueda.	

Curso alterno de los eventos	
Acción	**Curso alterno**
	6.1 Muestra las subespecies y/o formas contenidas dentro de la especie.
6.2 Selecciona la opción de ver el contenido de una subespecie o una forma.	
	6.3 Se inicia la Vista de Presentación Información.
Prioridad	Crítica

Mejoras			
Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación			
Sonido			
Texto			
Reglas Pedagógicas			

Tabla 8: Descripción textual de la Vista de Presentación ResultadosBúsqueda

Descripción textual de la Vista de Presentación ResultadosBúsqueda	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario conocer las coincidencias encontradas a partir una palabra o frase especificada como criterio de búsqueda.
Objetivos Pedagógicos	
Resumen	La vista se inicia cuando el usuario presiona la opción de buscar, el sistema muestra las coincidencias encontradas. La vista termina cuando el usuario escoge la opción de salir o cuando escoge alguna de las coincidencias.
Vistas asociadas	Vista de Presentación Información y Vista de Presentación Común, Vista de Presentación Principal (Subsistema Principal).
Referencias	R5

Precondiciones	Debe introducirse algún criterio de búsqueda, una palabra o frase.
Poscondiciones	Se muestran los resultados de la búsqueda.

Curso normal de eventos

Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Selecciona la opción de buscar.		
	2. Busca todas las coincidencias y las muestra. Permite seleccionar una opción para ver el contenido asociado al resultado de la búsqueda.	T_ResultadosB
3. Selecciona una opción para ver el contenido asociado.		
	4. Muestra el contenido asociado (Vista de Presentación Información).	

Curso alterno de los eventos

Acción	**Curso alterno**
	2.1 No existe ninguna coincidencia con el criterio de búsqueda.
3.1 Selecciona la opción de salir de la vista.	
	3.2 Termina la vista.
Prioridad	Crítica

Mejoras			
Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación			
Sonido			
Texto	T_ResultadosB	Texto que muestra un listado con todas las coincidencias encontradas como resultado de la búsqueda realizada por el sistema.	Existe.
Reglas Pedagógicas			

Tabla 9: Descripción textual de la Vista de Presentación Salir

Descripción textual de la Vista de Presentación Salir	
Actores de la Vista de Presentación	Usuario
Propósito	Permitir al usuario confirmar si realmente desea salir de la aplicación.
Objetivos Pedagógicos	
Resumen	La vista se inicia cuando el usuario selecciona la opción de salir de la aplicación. El sistema muestra un mensaje donde el usuario selecciona si realmente desea abandonar la aplicación o si desea mantenerse en ella.
Vistas asociadas	Subsistema Contenido, Vista de Presentación

	Principal y Vista de Presentación Historia.	
Referencias	R8	
Precondiciones		
Poscondiciones	Se ejecuta la Vista de Presentación Salir.	
Curso normal de eventos		
Acciones del actor	**Respuesta del sistema**	**Elementos de la vista**
1. Selecciona la opción de salir de la aplicación.		
	2. Se muestra un mensaje preguntando si realmente desea abandonar la aplicación y las opciones para confirmar y/o cancelar la operación.	T_Salir
3. Selecciona una de las opciones.		
	4. Si el usuario selecciona la opción de confirmación de salida del sistema se muestran los créditos y una vez culminados se cierra la aplicación, terminando así la vista de presentación.	A_Créditos
	5. Si se selecciona la opción de no salir de la aplicación se cierra la vista actual.	
Curso alterno de los eventos		

Acción		**Curso alterno**	
Prioridad		Crítica	
Mejoras			
Medias a utilizar			
Tipo de media	**Nombre**	**Descripción**	**Estado**
Imagen			
Video o Animación	A_Créditos	Animación que muestra los nombres de los desarrolladores y colaboradores implicados en la realización del producto.	Existe
Sonido			
Texto	T_Salir	Pregunta al usuario si realmente desea salir de la aplicación.	Existe
Reglas Pedagógicas			

2.9 Conclusiones del Capítulo

Se identificaron los requisitos funcionales y no funcionales que debe cumplir el sistema en desarrollo, así como también se definieron los conceptos fundamentales asociados, los que fueron utilizados para realizar el Modelo de Dominio. Además se definieron los subsistemas de los que está compuesto el producto y las Vistas de Presentación que componen cada uno de estos, describiendo textualmente las mismas como parte fundamental del capítulo. Se especificó la navegabilidad entre las vistas a través de los Diagramas de Estructura de Navegación.

CAPÍTULO 3: CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

Después de realizada la descripción de la solución propuesta se debe ir al siguiente paso: su construcción, por lo que el presente capítulo tiene como objetivo definir los artefactos necesarios para ello, y sus características. Los artefactos fundamentales que se generarán y que permitirán la construcción del producto son: Diagramas de Estructura de Presentación, Diagramas de Clases del Diseño, Diagramas de Interacción y el Modelo de Implementación.

3.2 Diagramas de Estructura de Presentación (DEP)

ApEM – L incorpora el Diagrama de Estructura de Presentación dentro de la Vista de Presentación definida por este lenguaje, que propone dos nuevas clases: la clase Estáticos y la clase Interacción. El objetivo fundamental de este diagrama es representar la estructura que tendrán las interfaces de comunicación con el usuario, “estableciendo una organización lógica de los elementos que las conforman, dejándole a los diseñadores gráficos la función de decidir dónde y cómo serán en términos visuales dichos elementos.” [11]

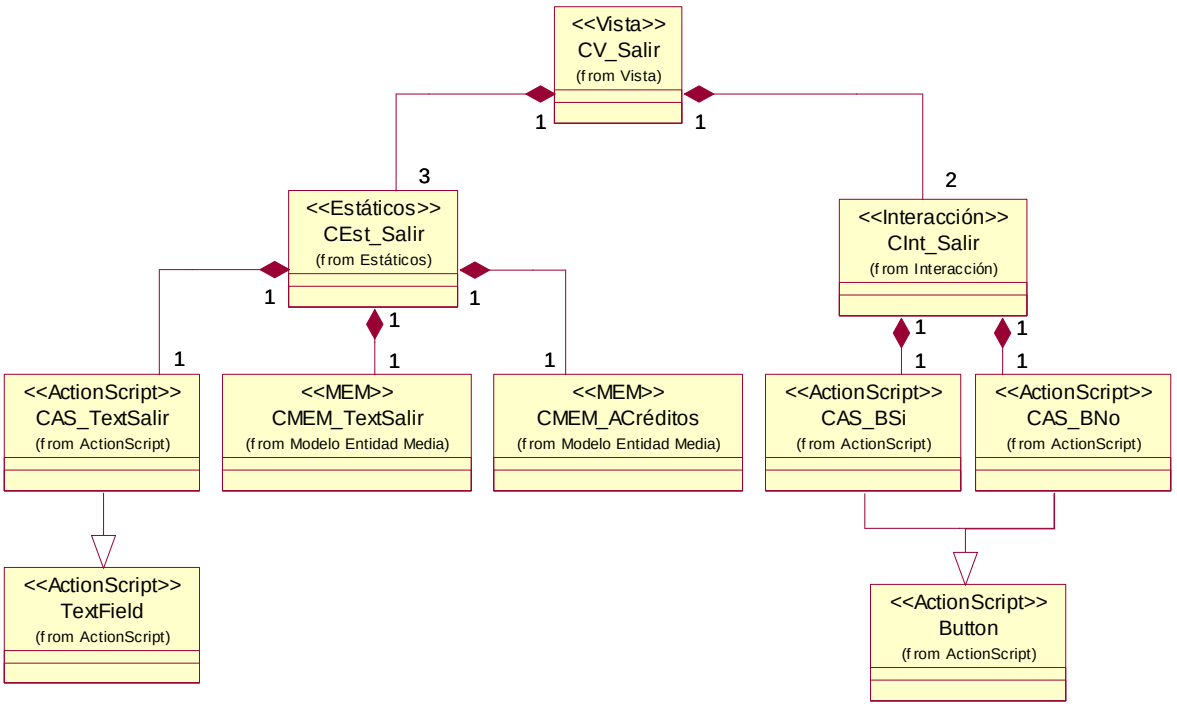


Figura 7: DEP Vista de Presentación Salir

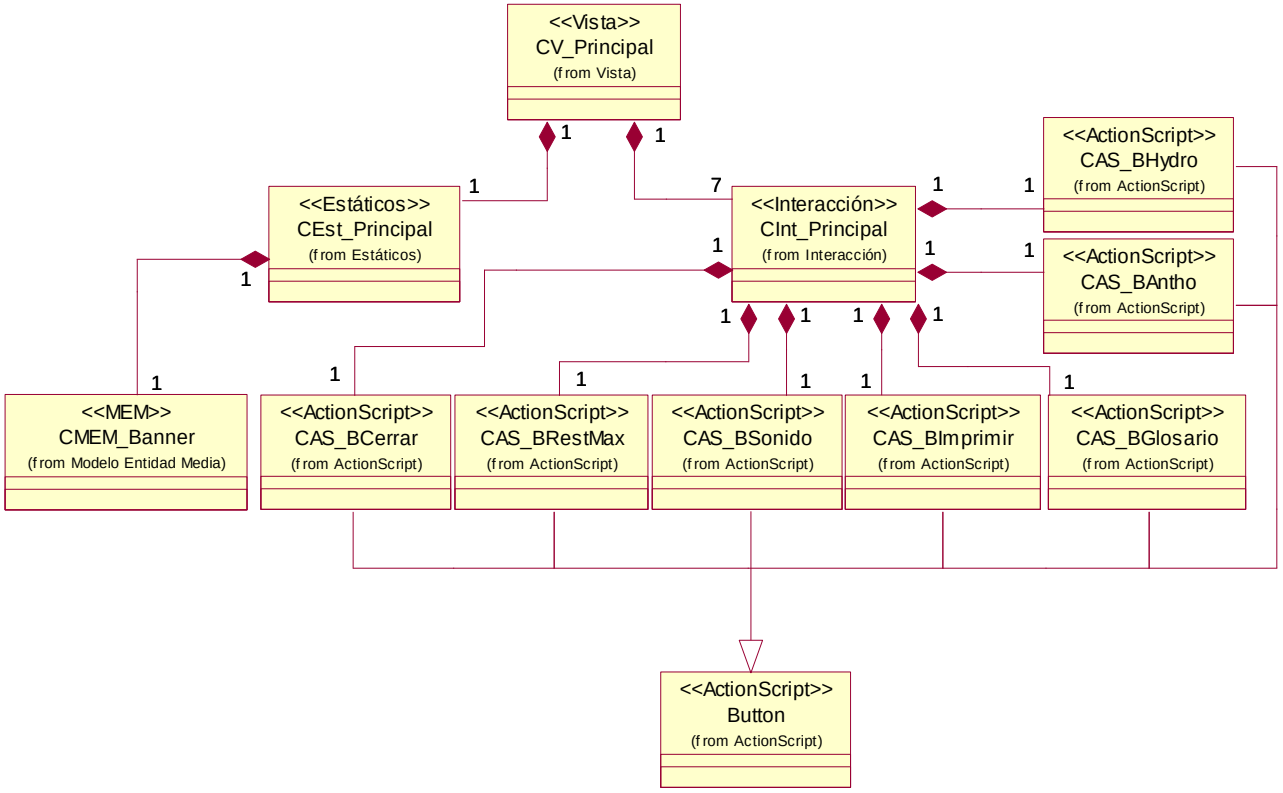


Figura 8: DEP Vista de Presentación Principal

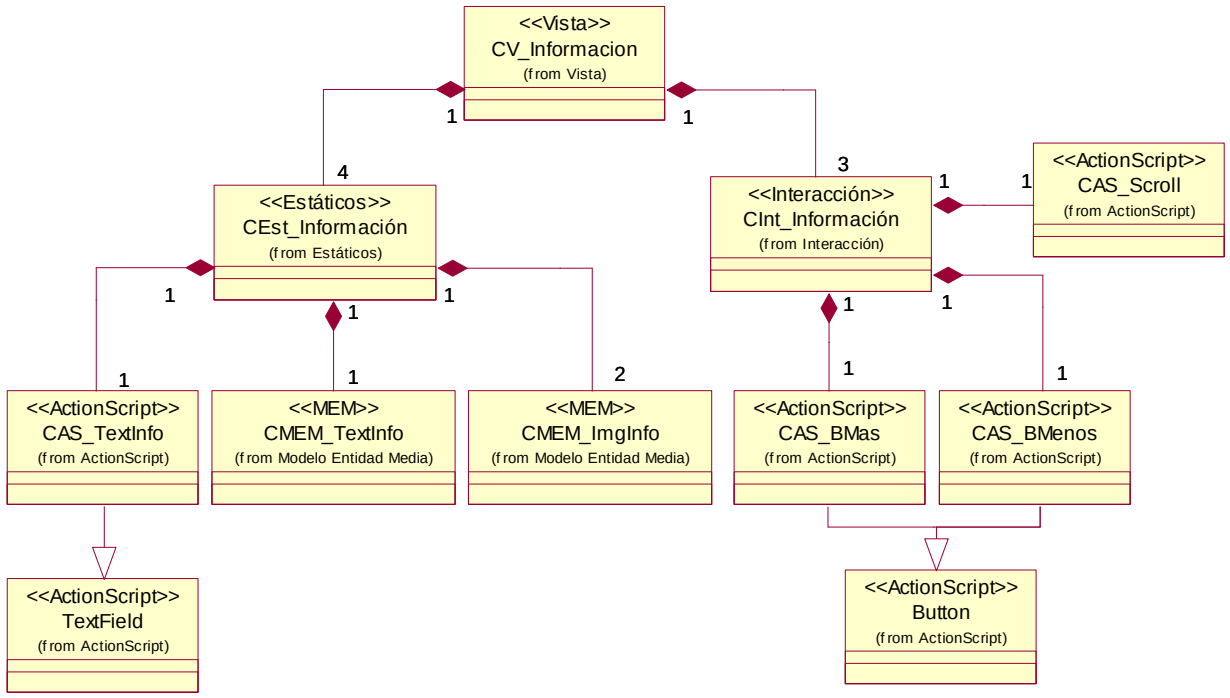


Figura 9: DEP Vista de Presentación Información

- DEP Vista Presentación [Ver Anexo 1]
- DEP Vista de Presentación Historia [Ver Anexo 2]
- DEP Vista de Presentación Mapa [Ver Anexo 3]
- DEP Vista de Presentación Galería [Ver Anexo 4]
- DEP Vista de Presentación Común [Ver Anexo 5]
- DEP Vista de Presentación ResultadosBúsqueda [Ver Anexo 6]

3.3 Modelo de Diseño

Con el objetivo de definir un producto con los suficientes detalles como para permitir su realización física se transita por una fase de diseño dentro del proceso de desarrollo de software, la cual consiste en aplicar distintos principios y técnicas que faciliten la implementación.

"El Modelo de Diseño es un modelo de objetos que describe la relación física… centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Además el Modelo de Diseño sirve de abstracción de la implementación del sistema y es, de ese modo, utilizado como una entrada fundamental de las actividades de la implementación." [5]

3.3.1 Diagramas de Clases

El Diagrama de Clases del Diseño es uno de los elementos agrupado en el área conceptual de UML de Estructura Estática, lo que ApEM – L mantiene aunque ha sufrido algunos cambios con la incorporación de algunos nuevos conceptos. En este diagrama las clases se describen, siempre teniendo en cuenta la tecnología en la que se implementará el sistema, con un mayor grado de detalle, debido precisamente a que constituyen una de las entradas fundamentales para la implementación.

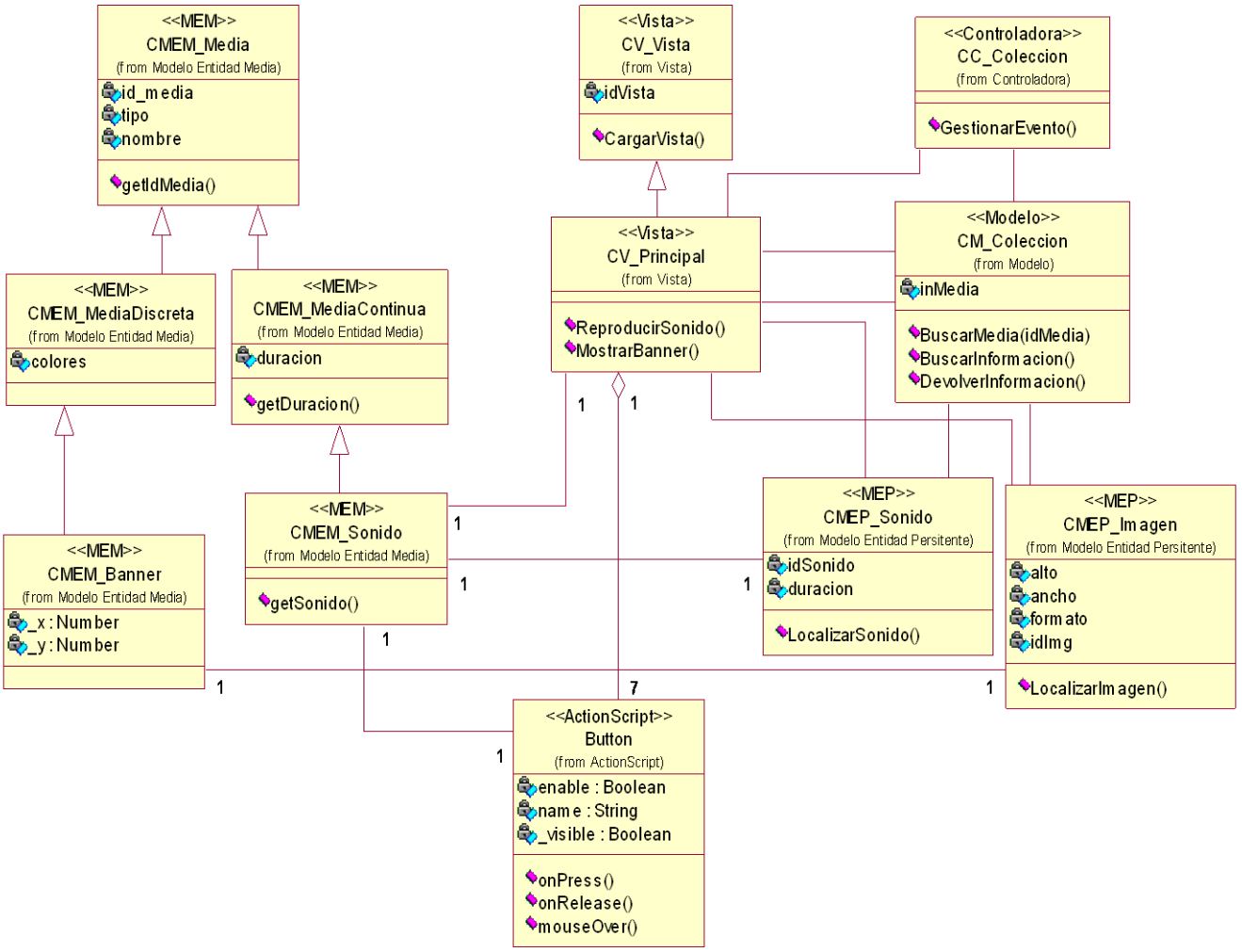


Figura 10: DCD Vista Principal

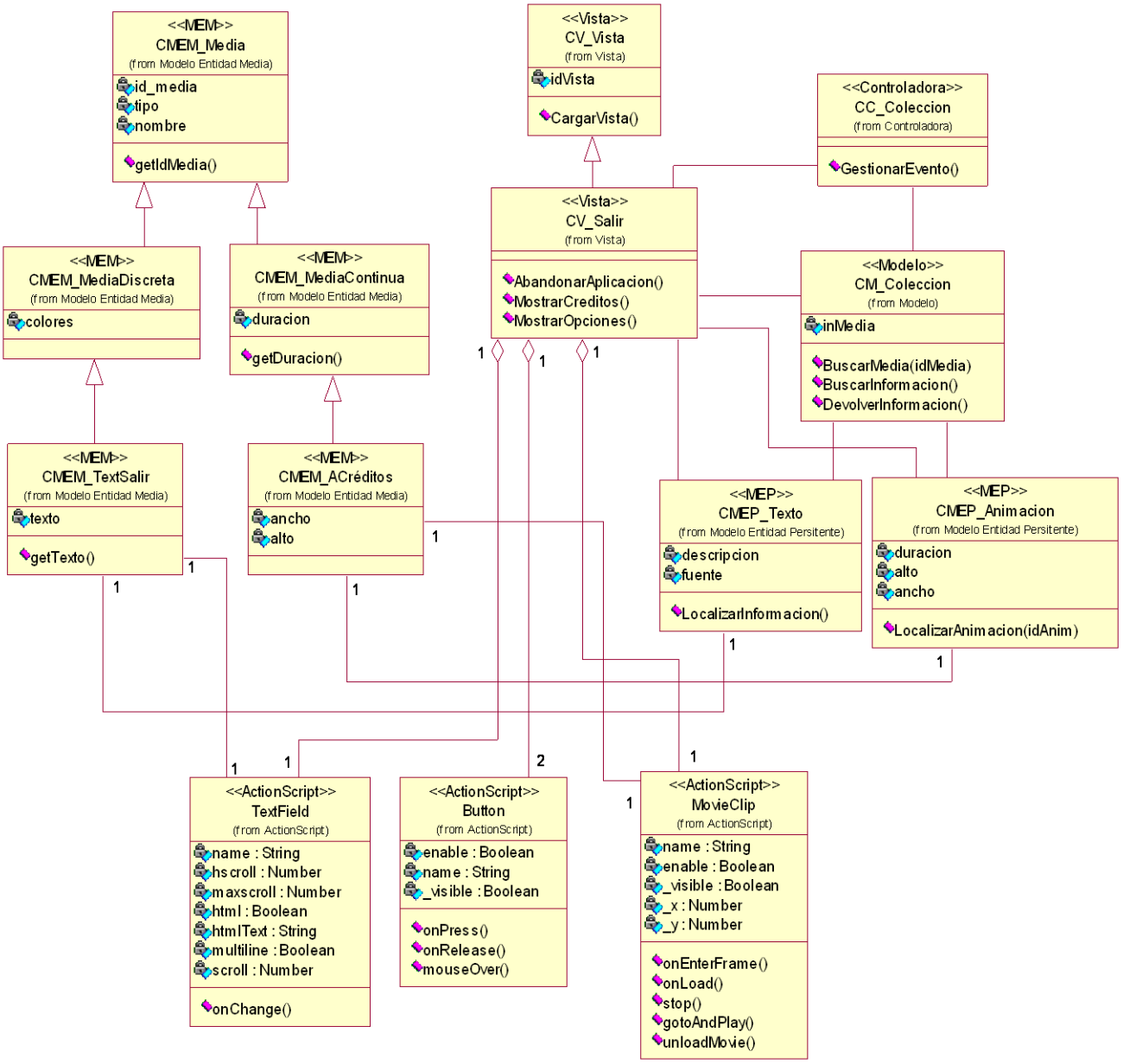


Figura 11: DCD Vista Salir

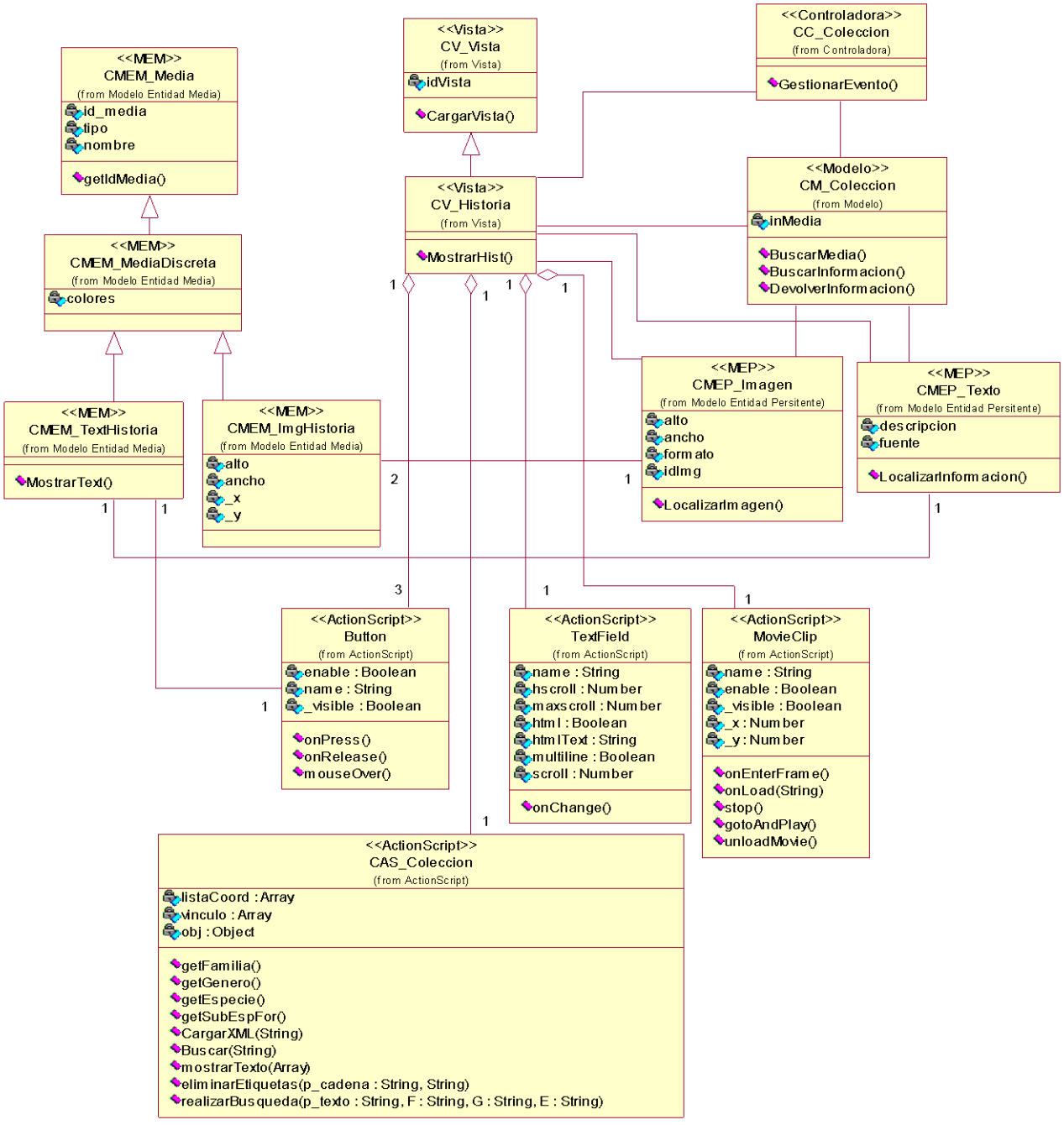


Figura 12: DCD Vista Historia

- Diagrama de clases Vista Común [Ver Anexo 7].
- Diagrama de clases Vista Galería [Ver Anexo 8].
- Diagrama de clases Vista Información [Ver Anexo 9].
- Diagrama de clases Vista Mapa [Ver Anexo 10].
- Diagrama de clases Vista Presentación [Ver Anexo 11].
- Diagrama de clases Vista ResultadosBúsqueda [Ver Anexo 12].

3.3.2 Diagramas de Interacción

"Una interacción es un conjunto de mensajes dentro de una colaboración que son intercambiados por roles de clasificador a través de roles de asociación. Cuando una colaboración existe en tiempo de ejecución, los objetos ligados a roles de clasificador intercambian instancias de mensajes a través de los enlaces ligados a los roles de asociación. Una interacción modela la ejecución de una operación, caso de uso, u otra entidad de comportamiento." [9]

ApEM – L como extensión de UML define dos tipos de diagramas de interacción, los de colaboración y los de secuencia. En el trabajo en desarrollo solo se realizarán los últimos que muestran los mensajes entre los objetos a través del paso del tiempo. (Ver Anexos 13 - 21).

3.4 Modelo de Implementación

El Modelo de Implementación contiene dentro de sí los Diagramas de Componentes y de Despliegue, los cuales describen los componentes a construir y su organización y la dependencia entre los nodos físicos en los que funcionará el sistema. Entre los componentes a construir podemos encontrar ejecutables, tablas de bases de datos, ficheros de código fuente, ficheros de código binario, scripts.

Los principales propósitos de la implementación son:

- Planificar las integraciones de sistema necesarias en cada iteración.
- Distribuir el sistema asignando componentes ejecutables a nodos en el Diagrama de Despliegue.
- Implementar las clases y subsistemas encontrados durante el Diseño. En particular, las clases se implementan como componentes de fichero que contienen código fuente.

- Probar los componentes individualmente y a continuación integrarlos compilándolos y enlazándolos en uno o más ejecutables antes de ser enviados para ser integrados y llevar a cabo las comprobaciones del sistema. [5]

3.4.1 Diagramas de Componentes

Un diagrama de componentes muestra las relaciones y la organización entre componentes de software, ya sean de código fuente, binarios o ejecutables. Para su construcción se tienen en cuenta los requisitos relacionados con la facilidad de desarrollo, la gestión de software, la reutilización y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. El diagrama de componentes general muestra el subsistema Principal y su relación con el archivo XML historia, así como la relación entre el subsistema Contenido y los archivos XML colección, pieImagen y glosario. Ambos subsistemas se relacionan el paquete Clases.

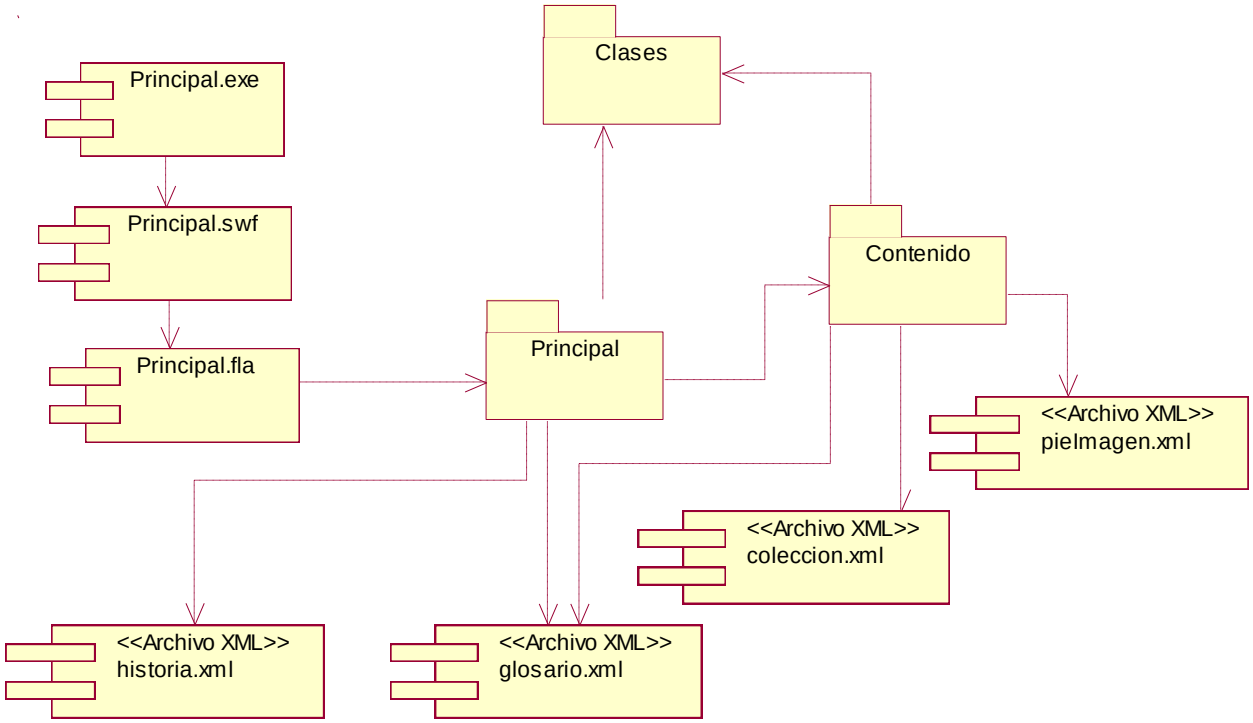


Figura 13: Diagrama de componentes general

- Diagrama de componentes Paquete Clases [Ver Anexo 22].
- Diagrama de componentes Paquete Principal [Ver Anexo 23].
- Diagrama de componentes Paquete Contenido [Ver Anexo 24].

3.4.2 Diagrama de Despliegue

El diagrama de despliegue describe la distribución física del sistema mediante un modelo de objetos, es decir es una representación gráfica de los nodos de cómputo y muestra como estos distribuyen las funcionalidades. Es utilizado como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño.

En el caso de estudio presente, el diagrama está compuesto por dos únicos nodos, uno representando la computadora en donde se ejecuta la aplicación, la cual solo debe cumplir con los requisitos mínimos expuestos como especificaciones no funcionales y otro nodo representando una impresora como dispositivo externo para dar cumplimiento al requisito funcional de poder imprimir contenido expuesto en la aplicación.

Figura 14: Diagrama de Despliegue

3.5 Descripción de Archivos XML

Toda la información que se mostrará en la aplicación será almacenada en archivos XML debido a las facilidades que este lenguaje ofrece en el trabajo con Flash. Para el desarrollo de la aplicación se utilizaron tres archivos XML, uno para almacenar el contenido y las rutas de las imágenes de la historia de la colección y la descripción de las mismas, otro para las descripciones en español y francés así como las rutas de las imágenes de las especies, las formas y/o subespecies y un tercero para almacenar las descripciones de las imágenes de la colección.

3.5.1 Archivo XML historia

```
<!ELEMENT historia (descripcion, fotos)+>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT fotos (img)+>
<!ELEMENT img (#PCDATA)+>

<!ATTLIST img id CDATA #REQUIRED>
<!ATTLIST img dir CDATA #REQUIRED>
```

3.5.2 Archivo XML colección

```
<!ELEMENT contenido (clase)+>
<!ELEMENT clase (orden)+>
<!ELEMENT orden (familia)+>
<!ELEMENT familia (genero)+>
<!ELEMENT genero (especie)+>
<!ELEMENT especie (descripcion, ((forma?, subespecie?)+ | imgs))+>
<!ELEMENT forma (descripcion, imgs)+>
<!ELEMENT subespecie (descripcion, imgs)+>
<!ELEMENT imgs (mn, coleccion, mapas)+>
<!ELEMENT mn (img)?>
<!ELEMENT coleccion (img)?>
<!ELEMENT mapas (mapa)+>
<!ELEMENT img EMPTY>
<!ELEMENT mapa EMPTY>
<!ELEMENT descripcion (esp, fra)>
<!ELEMENT esp (#PCDATA)>
<!ELEMENT fra (#PCDATA)>

<!ATTLIST clase id CDATA #REQUIRED>
<!ATTLIST clase nombre CDATA #REQUIRED>

<!ATTLIST orden id CDATA #REQUIRED>
<!ATTLIST orden nombre CDATA #REQUIRED>

<!ATTLIST familia id CDATA #REQUIRED>
<!ATTLIST familia nombre CDATA #REQUIRED>

<!ATTLIST genero id CDATA #REQUIRED>
<!ATTLIST genero nombre CDATA #REQUIRED>

<!ATTLIST especie id CDATA #REQUIRED>
<!ATTLIST especie nombre CDATA #REQUIRED>
<!ATTLIST especie forsub (si | no) #REQUIRED>
```

<!ATTLIST forma id CDATA #REQUIRED>
<!ATTLIST forma nombre CDATA #REQUIRED>

<!ATTLIST img id CDATA #REQUIRED>
<!ATTLIST img dir CDATA #REQUIRED>

<!ATTLIST subespecie id CDATA #REQUIRED>
<!ATTLIST subespecie nombre CDATA #REQUIRED>

3.5.3 Archivo XML pielmagen

<!ELEMENT dataroot (Especies_x0020_con_x0020_fotos)+>
<!ELEMENT Especies_x0020_con_x0020_fotos (Id, IDO_x0020__x0023__x0020_Catalográfico, Existe_x0020_foto_x003F_, Nombre_x0020_jpg, FAMILIA, GENERO, ESPECIE, AUTOR_x0020_Y_x0020_AÑO, FECHA_x0020_COL, LATITUD, LONGITUD, LOCALIDAD, PROF, COLECTOR, DETERMINADOR)+>

<!ELEMENT Id (#PCDATA)+>
<!ELEMENT IDO_x0020__x0023__x0020_Catalográfico (#PCDATA)+>
<!ELEMENT Existe_x0020_foto_x003F_ (#PCDATA)+>
<!ELEMENT Nombre_x0020_jpg (#PCDATA)+>
<!ELEMENT FAMILIA (#PCDATA)+>
<!ELEMENT GENERO (#PCDATA)+>
<!ELEMENT ESPECIE (#PCDATA)+>
<!ELEMENT AUTOR_x0020_Y_x0020_AÑO (#PCDATA)+>
<!ELEMENT FECHA_x0020_COL (#PCDATA)+>
<!ELEMENT LATITUD (#PCDATA)+>
<!ELEMENT LONGITUD (#PCDATA)+>
<!ELEMENT LOCALIDAD (#PCDATA)+>
<!ELEMENT PROF (#PCDATA)+>
<!ELEMENT COLECTOR (#PCDATA)+>
<!ELEMENT DETERMINADOR (#PCDATA)+>

3.5.4 Archivo XML Glosario

<!ELEMENT glosario (letra)+>
<!ELEMENT letra (palabra)+>
<!ELEMENT palabra (#PCDATA)>

<!ATTLIST letra let CDATA #REQUIRED>

3.6 Conclusiones del Capítulo

Se obtuvieron los artefactos fundamentales correspondientes a los flujos de trabajo de diseño e implementación: Modelo de Diseño y Modelo de Implementación, así como los Diagramas de Estructura de Presentación que fueron definidos por el lenguaje de modelado para aplicaciones con tecnología multimedia. Se realizó una descripción de los archivos XML utilizados para el tratamiento de datos.

CAPÍTULO 4: ESTUDIO DE FACTIBILIDAD

4.1 Introducción

La estimación de costo de un software forma una parte importante y necesaria en el desarrollo del mismo. Por lo general se realiza en la etapa inicial del proyecto cuando lo que se tiene es a lo sumo una visión general de este. Estudios realizados demuestran que una mala estimación da como resultados atrasos en la entrega del producto, costos por encima de lo planificado y riesgo de que el producto a entregar no cumpla con el objetivo para el que fue construido. De dicha experiencia viene la necesidad de estimar correctamente los esfuerzos de un proyecto.

Con una correcta estimación no solo se obtiene el costo, también la estimación de los resultados del proyecto y los valores de tiempo y recursos requeridos y selección de métodos, procesos, herramientas, normas, organización.

La estimación tiene como objetivo determinar la posibilidad real de llevar a cabo el desarrollo de un proyecto, es decir, el estudio de factibilidad de acuerdo a las diferentes restricciones.

Las restricciones están dadas por las características propias del proyecto y/o grupo de trabajo:

- ✓ Organizativas: por la estructura, procesos y personas.
- ✓ Económicas: se analiza la relación costo-beneficio.
- ✓ Técnicas: referente a las habilidades que posee el grupo de trabajo, experiencia con que cuenta y los recursos disponibles.
- ✓ Tiempo: Se tiene en cuenta cuando existen requerimiento de fecha de cumplimiento.

Existen varios métodos de estimación para definir la factibilidad de un proyecto, entre los más conocidos y utilizados están los métodos Análisis por puntos de casos de uso y Análisis de puntos de función y COCOMO II (Constructive Cost Model).

Se abordará en el siguiente capítulo el estudio de la factibilidad realizado para el proyecto utilizando como método de estimación: Análisis de puntos de función y COCOMO II.

4.2 Análisis de puntos de función y COCOMO II

Hoy en día metodologías ampliamente difundidas, como el Proceso Unificado de Rational (RUP) ha demostrado que la especificación de los requerimientos mediante Casos de Uso es uno de los métodos más efectivos para capturar la funcionalidad de un sistema. Aunque los Casos de Uso permiten especificar la funcionalidad de un sistema bajo análisis, no permiten por sí mismos efectuar una estimación del tamaño que tendrá el sistema o del esfuerzo que tomaría implementarlo. Para la estimación de un proyecto a partir de sus requerimientos, una de las técnicas más utilizadas es el Análisis de Puntos de Función. Esta técnica cuantifica en Puntos de Función el tamaño de un sistema que serían unidades independientes del lenguaje de programación, tecnologías utilizadas, metodologías y/o plataformas.

Desde hace algunos años el SEI (Software Engineering Institute) propone un método para la estimación del esfuerzo llamado COCOMO II. Éste método está basado en ecuaciones matemáticas que permiten calcular el esfuerzo a partir de ciertas métricas de tamaño estimado, como el Análisis de Puntos de Función y las líneas de código fuente (SLOC, Source Line Of Code).

En relación a los Puntos de Función, las transacciones se clasifican de la siguiente manera:

- ✓ Entradas Externas (EI)
- ✓ Salidas Externas (EO)
- ✓ Consultas Externas (EQ)

En relación a los Puntos de Función, los archivos se clasifican de la siguiente manera:

- ✓ Archivos Lógicos Internos (ILF)
- ✓ Archivos de Interfaz Externos (EIF)

Entradas Externas: No hay

Tabla 9: Salidas Externas

Nombre de la salida externa	**Cantidad de archivos referenciados**	**Cantidad de elementos de datos**	**Clasificación (Baja, Media, Alta)**
Mostrar el contenido “Historia”.	1	1	Baja
Mostrar imágenes referentes a la	1	1	Baja

historia de la colección.			
Mostrar el contenido “Información”	1	1	Baja
Mostrar imágenes referentes al medio natural de una especie	1	1	Baja
Mostrar imágenes referentes a una especie en la colección	2	1	Baja
Mostrar imagen del mapa donde se ubica una especie	1	1	Baja
Mostrar el contenido “Glosario”	1	1	Baja
Mostrar contenido “Clase Anthozoa”	1	1	Baja
Mostrar contenido “Clase Hydrozoa”	1	1	Baja
Reproducir música de fondo	1	1	Baja

Tabla 10: Consultas Externas

Nombre de la consulta externa	**Cantidad de archivos referenciados**	**Cantidad de elementos de datos**	**Clasificación (Baja, Media, Alta)**
Realizar búsqueda de datos.	1	1	Baja

Tabla 11: Ficheros lógicos internos

Nombre del fichero interno	**Tipos de registro**	**Cantidad de elementos de datos**	**Clasificación (Baja, Media, Alta)**
Colección	1	5	Baja
Pie-imagen	1	12	Baja
Historia	1	2	Baja
Glosario	1	1	Baja

La información asociada al fichero interno colección es: clase, orden, familia, género y especie. De la misma forma el fichero interno pie-imagen tiene asociado información de: nombre, familia, género,

especie, autor, fecha, latitud, longitud, localidad, profundidad, colector y determinador. Por su parte el archivo interno historia solo cuenta con: descripción e imágenes y glosario solo tiene la información asociada a los términos.

Tabla 12: Puntos de Función

	Total	**Complejidad**						**Aporte**	
		Simple	**Valor**	**Media**	**Valor**	**Compleja**	**Valor**		
Salidas Externas	10	10	4	-		-		40	
Consultas Externas	1	1	3	-		-		3	
Archivo lógicos internos	4	4	3	-		-		12	
								55	**Total**

Teniendo en cuenta la suma total de los aportes de todos los elementos se obtienen los Puntos de Función sin ajustar:

UFP (Puntos de Función sin ajustar) = 40 + 3 + 12 = 55

Una vez obtenidos los Puntos de Función sin ajustar se estima el esfuerzo a través del método COCOMO II, el cual se puede aplicar directamente sobre los Puntos de Función sin ajustar y es uno de los más utilizados en la actualidad para la estimación del esfuerzo cuando no se tiene información histórica a la cual recurrir.

De forma general, COCOMO II está basado en dos modelos: uno aplicable al comienzo de los proyectos (Diseño preliminar, en inglés Early Design) y otro aplicable luego del establecimiento de la arquitectura del sistema (Post arquitectura, en inglés Post Architecture).

Se utilizará el modelo de Diseño preliminar (Early Design) ya que en esta etapa no se tiene suficiente conocimiento del proyecto como para hacer una estimación delicada. También se tienen en cuenta la exploración de arquitecturas alternativas del sistema y los conceptos de operación. A partir de ahí el modelo propone la utilización de Puntos de Función como medida de tamaño y un conjunto de 7 factores (cost drivers) que afectan al esfuerzo del proyecto. Estos 7 factores son agrupaciones de los factores que se utilizan en la otra variante del modelo (Post Arquitectura).

Características	Valor
Puntos de función desajustados	55
Lenguaje	ActionScript 2.0
Instrucciones fuentes por puntos de función(SLOC/UFP)	66
Instrucciones fuentes	3630

4.3 Estimación del esfuerzo, cantidad de hombres, tiempo de desarrollo y costo

Para aplicar la ecuación de cálculo del esfuerzo nominal se necesita por un lado convertir los puntos de función sin ajustar a KSLOC (Source Lines Of Code, en miles), y por otro calcular el factor escalar B de acuerdo a las características del proyecto. La estimación del esfuerzo se realiza tomando como base la siguiente ecuación:

$$\mathbf{PM_{nominal} = A \times (Size)^{B}}$$

Donde:

$PM_{nominal}$: es el esfuerzo nominal requerido en meses-hombre.

Size: es el tamaño estimado del software, en miles de líneas de código (KSLOC) o en Puntos de Función sin ajustar (convertibles a KSLOC mediante un factor de conversión que depende del lenguaje y la tecnología).

A: es una constante que se utiliza para capturar los efectos multiplicativos en el esfuerzo requerido de acuerdo al crecimiento del tamaño del software. El modelo la calibra inicialmente con un valor de 2.94.

B: es una constante denominada Factor escalar, la cual tiene un impacto exponencial en el esfuerzo y su valor está dado por la resultante de los aspectos positivos sobre los negativos que presenta el proyecto.

Cálculo del Factor Escalar (B)

$$\mathbf{B = 0.91 + 0.01 \times \Sigma (SF_i)}$$

SF_i= valor de la variable escalar

Variables

- ✓ PREC, variable de precedencia u orden secuencial del desarrollo.
- ✓ FLEX, variable de flexibilidad del desarrollo.
- ✓ RSEL, indica la fortaleza de la arquitectura y métodos de estimación y reducción de riesgos.
- ✓ TEAM, esta variable refleja la cohesión y madurez del equipo de trabajo.
- ✓ PMAT, relaciona el proceso de madurez del software

Tabla 13: Variables escalares

Variable	**Descripción**	**Nivel de cuantificación(Muy bajo, Bajo, Nominal, Alto, Muy alto y Extra alto)**	**Valores**
PREC	Se tiene algo de precedencia en el trabajo con este tipo de sistemas.	Nominal	3.65
FLEX	Algo de relajación en cuanto a la flexibilidad del desarrollo.	Nominal	3.04
RSEL	No todos los riesgos se han mitigado	Nominal	4.12
TEAM	La interacción del equipo de desarrollo es altamente cooperativa.	Muy Alto	1.10
PMAT	La madurez del	Baja	6.24

	proceso de software es baja.		

$\Sigma SF_i = 3.65 + 3.04 + 4.12 + 1.10 + 6.24 = 18.15$

$B = 0.91 + 0.01 * 18.15 = 1.0915$

A = tomamos el valor por defecto del modelo = 2.94

Size: se calcula como el producto de los puntos de función sin ajustar por un factor de conversión que depende del lenguaje a utilizar en el desarrollo del sistema.

Size= 55*66 = 3630 SLOC = 3.6 KSLOC

$PM_{nominal} = 2.94 * (3.6)^{1.0915}$

$PM_{nominal} = 2.94 * 4.047 = 11.8981$

Para completar la estimación, hay que ajustar el esfuerzo nominal de acuerdo a las características del proyecto.

$PM = PM_{nominal} \times \pi (ME_i)$

Donde:

Los **ME_i** (multiplicadores de esfuerzo) varían en función del modelo de estimación seleccionado (Diseño Preliminar o Post arquitectura) y representan las características del proyecto que expresan su repercusión en el desarrollo total del producto.

Multiplicadores de esfuerzo:

- ✓ PERS: Capacidad del personal.
- ✓ RCPX: Complejidad del producto.
- ✓ RUSE: Reusabilidad.
- ✓ PDIF: Dificultad de la plataforma.
- ✓ PREX: Experiencia del personal.

- ✓ SCED: Calendario.
- ✓ FCIL: Facilidades.

Tabla 14: Multiplicadores de esfuerzo

Multiplicador	**Descripción**	**Nivel de cuantificación (Extra bajo, Muy bajo, Bajo, Nominal, Alto, Muy alto y Extra alto)**	**Valores**
PERS	Se tienen analistas y programadores con alta capacidad de trabajo en equipo y eficientes.	Nominal	1
RCPX	Las exigencias de confiabilidad, documentación y volumen de datos son moderados, y la complejidad del producto es simple.	Nominal	1
RUSE	Se pretende reutilizar el código para futuros productos en el proyecto.	Nominal	1
PDIF	No existen restricciones en cuanto al tiempo del CPU o al consumo de memoria, la plataforma es muy estable.	Baja	0.87
PREX	El personal del equipo de	Nominal	1

	trabajo, dígase analistas y programadores tienen aproximadamente 1 año de experiencia en el trabajo con las herramientas y el lenguaje utilizado.		
SCED	Se requiere terminar el proyecto en el tiempo estimado.	Alto	1
FCIL	Se tienen herramientas CASE simples e infraestructura de comunicación básica	Bajo	1.10

ME = 1*1*1*0.87*1*1*1.10= 0.957

$\pi (ME_i)$= 0.957

PM = 11.8981* 0.957= 11.38

PM = 11.38 meses- hombres

Ya encontrado el esfuerzo (PM), se aplican algunas fórmulas de Boehm para calcular el tiempo de desarrollo de la aplicación (TDEV).

$TDEV= 3.67*(PM)^{0.28+0.002* \Sigma SFi}$

$TDEV=3.67 * (11.38)^{0.28+0.002* 18.15} = 3.67 * (11.38)^{0.3163}$

TDEV= 7.9199 = 8 meses

Para estimar cuántas personas requiere el desarrollo:

CH= PM/TDEV

CH= 11.38 / 8 = 1.422

CH= 2 hombres

Costo:

Costo = CHM * PM

CHM= CH* Salario mínimo

Considerando que el producto es desarrollado por estudiantes de 5to año de la Universidad de las Ciencias Informáticas, el salario mínimo a considerar es de $100.

CHM = 2*100 = 200

Costo = 200* 11.38 = 2276

Costo = $ 2276

4.4 Beneficios tangibles

- ✓ Documentación necesaria para implementar la aplicación.
- ✓ Reusabilidad de código.
- ✓ Multimedia Colección Corales Pétreos.

4.5 Beneficios intangibles

Los beneficios intangibles asociados al desarrollo del software con tecnología multimedia “Colección Corales Pétreos” son los siguientes:

- ✓ Se podrán realizar estudios de la Colección de Corales Pétreos existente en Cuba sin necesidad de trasladarlos de su lugar de conservación.
- ✓ Ayudará a reducir en un alto por ciento el deterioro de la colección prolongando así sus años de vida.

4.6 Análisis de costo y beneficio

Para desarrollar la aplicación con tecnología multimedia “Colección Corales Pétreos” no es necesario grandes gastos de recursos en tecnología de punta, ni sistemas de cómputo de última generación de grandes capacidades de procesamiento y almacenamiento. La aplicación es realizada para usuarios con

baja o ninguna experiencia en el uso de computadoras por lo que posee una interfaz amigable, de fácil uso y comunicación. Su mayor aporte es la ayuda a la reducción del deterioro de la colección de corales pétreos. Según los datos obtenidos a través de la estimación, y analizando la relación costo-beneficios del producto, se decidió que si era factible desarrollarlo, ya que el mismo ayudará al estudio de la colección de corales pétreos de forma digital haciendo posible su mejor conservación y de esta forma ahorrando gastos al país y al Acuario Nacional de Cuba cubriendo así el costo de la producción ($2276).

4.7 Conclusiones del capítulo

Se determinó que es factible desarrollar el producto que debe estar listo según el tiempo de desarrollo estimado en 8 meses, con el trabajo de 2 hombres y un costo aproximado de 2276 pesos en moneda nacional. Se establecieron los beneficios que se obtendrán del desarrollo de esta aplicación, beneficio para la Colección de Corales Pétreos, para el Acuario Nacional de Cuba y por lo tanto para el país.

CONCLUSIONES

- Con el objetivo de lograr un óptimo desarrollo se llevó a cabo un estudio sobre la ingeniería de software aplicada a productos multimedia, seleccionando como metodología de desarrollo el Proceso Unificado de Rational (RUP) y como lenguaje de modelado ApEM-L.
- Se investigó sobre las posibles herramientas para el desarrollo de aplicaciones con tecnología multimedia y se seleccionó Macromedia Flash en su versión 8.0.
- Se llevo a cabo el análisis, el diseño y la implementación de la aplicación con tecnología multimedia “Colección de Corales Pétreos” obteniendo los artefactos y la documentación necesaria para garantizar el futuro mantenimiento del sistema.
- Se obtuvo un producto interactivo que permitirá realizar estudios de la Colección de Corales Pétreos existente en Cuba sin necesidad de trasladarlos de su lugar de conservación y ayudará a reducir en un alto por ciento el deterioro de la colección prolongando sus años de vida.

RECOMENDACIONES

- Difundir el contenido de la aplicación a la comunidad científica o a cualquier interesado en el estudio de la Colección de Corales Pétreos tanto dentro como fuera de Cuba.

- Ampliar las funcionalidades del producto obtenido brindando la posibilidad de observar con más detalles las imágenes de las especies para un mejor estudio de estas, al igual que los mapas que muestran la distribución geográfica de las mismas.

REFERENCIAS BIBLIOGRÁFICAS

[1]: Diccionario de la Real Academia Española Vigésima segunda edición [En línea]. Disponible en: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=tecnología [Citado el 16/02/2009].

[2]: Diccionario de la Real Academia Española Vigésima segunda edición [En línea]. Disponible en: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=multimedia [Citado el 16/02/2009].

[3]: Diccionario de la Real Academia Española Vigésima segunda edición [En línea]. Disponible en: http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=metodología [Citado el 18/02/2009].

[4]: PÉREZ, Y. *Metodologías para el desarrollo de software educativo: Un estudio comparativo.* Universidad de las Ciencias Informáticas (UCI), Ciudad de la Habana, Cuba. 2006.

[5]: BOOCH, G., JACOBSON, I. y RUMBAUGH, J. *El Proceso Unificado de Desarrollo de Software,* Madrid, Editorial Pearson Educación, S. A., 2000.

[6]: IBM Corporation 1987, 2005, Rational Unified Process Versión 7.0.1, Ayuda extendida de Rational Rose en español.

[7]: BECK, K. *Extreme Programming Explained: Embrace Change*. Editorial Addison Wesley Longman, 2000.

[8]: CALERO, M. Una explicación de la programación extrema (XP). V Encuentro usuarios xBase 2003 MADRID [En línea]. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf [Citado el 25/02/2009].

[9]: BOOCH, G., JACOBSON, I. y RUMBAUGH, J. *El Lenguaje Unificado de Modelado. Manual de Referencia* (1ra ed.). Madrid, Editorial Addison-Wesley. 2000.

[10]: SAUER, S. y ENGELS, G. *Extending UML for Modeling of Multimedia Applications* [En línea]. Disponible en http://www.itec.uni-klu.ac.at/~harald/proseminar02/sauer1.pdf. 2001. [Citado el 5/02/2009]

[11]: MSc. CIUDAD, F. *ApEM-L como una nueva solución a la modelación de aplicaciones educativas multimedia en la UC.* Universidad de las Ciencias Informáticas (UCI), Ciudad de la Habana. 2007.

[12]: GARCIA, M. Herramientas de Autor para la enseñanza. [En línea]. Disponible en: http://www.telecable.es/personales/mmggg/hdeautor/ [Citado el 25/02/2009].

[13]: Programas muy gratis. [En línea]. Disponible en: http://muygratis.wordpress.com/page/2/ [Citado el 25/02/2009].

[14]: Curso Online Flash 8 [En línea]. Disponible en: http://www.aulaclic.es/flash8/t_1_1.htm Marzo 2006 [Citado el 5/02/2009].

[15]: Red de software. [En línea]. Disponible en: http://espanol.softpicks.net/software/Macromedia-Director_es-54371.htm [Citado el 26/02/2009].

[16]: Director. Febrero 2009. [En línea]. Disponible en http://seminarioarely.blogspot.com/2009_01_01_archive.html [Citado el 1/03/2009].

[17]: Mediator 6 para aplicaciones multimedia. [En línea]. Disponible en: http://www.pri.jovenclub.cu/jc/pa/tutoriales/Mediator.doc. [Citado el 1/03/2009]

[18]: Quersystem, Tecnología. [En línea]. Disponible en: http://www.quersystem.com/index.php?option=com_content&view=article&id=20&Itemid=3 [Citado el 3/03/2009].

[19]: Curso Online Flash 8 [En línea]. Disponible en: http://www.aulaclic.es/flash8/t_17_1.htm Marzo 2006 [Citado el 5/03/2009].

[20]: DRUCKER, P. The Discipline of Innovation. Harvard Business Review. Mayo – Junio 1985.

[21]: BAUMEISTIER, H., KOCH, N. y MANDEL, L. *Towars a UML Extension for Hypermedia Design*. [En línea]. Disponible en: http://citeseer.ist.psu.edu/cache/papers/cs/31048/http:zSzzSzwww.fast.dezSzProjektezSzforsoftzSzuml99zSzuml99.pdf/baumeister99towards.pdf. 2001. [citado el 20/03/2009].

BIBLIOGRAFÍA

1. BAUMEISTIER, H., KOCH, N. y MANDEL, L. *Towars a UML Extension for Hypermedia Design*. [En línea]. Disponible en: http://citeseer.ist.psu.edu/cache/papers/cs/31048/http:zSzzSzwww.fast.dezSzProjektezSzforsoftzSzuml99zSzuml99.pdf/baumeister99towards.pdf. 2001. [Citado el 3/02/2009].
2. BECK, K. *Extreme Programming Explained: Embrace Change*. Editorial Addison Wesley Longman, 2000.
3. BOOCH, G., JACOBSON, I. y RUMBAUGH, J. *El Lenguaje Unificado de Modelado. Manual de Referencia* (1ra ed.). Madrid, Editorial Addison-Wesley. 2000.
4. BOOCH, G., JACOBSON, I. y RUMBAUGH, J. *El Proceso Unificado de Desarrollo de Software,* Madrid, Editorial Pearson Educación, S. A., 2000.
5. CALDERÓN, A. y otros. *Metodologías Ágiles.* Universidad Nacional de Trujillo, Perú. 2007.
6. CALERO, M. Una explicación de la programación extrema (XP). V Encuentro usuarios xBase 2003 MADRID [En línea]. Disponible en: http://www.willydev.net/InsiteCreation/v1.0/descargas/prev/explicaxp.pdf [Citado el 25/02/2009].
7. CORRALES, C*. LA TECNOLOGIA MULTIMEDIA: Una Nueva Tecnología de Comunicación e Información. Características, concepciones y aplicaciones.* [En línea]. Disponible en: http://iteso.mx/%7Ecarlosc/pagina/documentos/multidef.htm. 1994. [Citado el 5/02/2009].
8. Curso Online Flash 8 [En línea]. Disponible en: http://www.aulaclic.es/index.html Marzo 2006 [Citado el 5/03/2009].
9. DELGADO, E. Revista de arquitectura e ingeniería. Artículo 2 Metodologías de desarrollo de software ¿Cuál es el camino? [En línea]. Disponible en: http://www.empai-matanzas.co.cu/revista/REVISTA_archivos/Page996.htm [Citado el 25/02/2009].
10. DRUCKER, P. The Discipline of Innovation. Harvard Business Review. Mayo – Junio 1985.
11. GARCIA, M. Herramientas de Autor para la enseñanza. [En línea]. Disponible en: http://www.telecable.es/personales/mmggg/hdeautor/ [Citado el 25/02/2009].

12. IBM Corporation 1987, 2005, Rational Unified Process Versión 7.0.1, Ayuda extendida de Rational Rose en español.

13. LIC. VALDÉS, M., Lic. Briggs, M., Ing. Gui, Y. Empleo de una multimedia en el desarrollo de dos conferencias del programa de química del Curso Premédico de la ELAM. [En línea]. Disponible en http://www.informaticahabana.com/evento_virtual/files/EDU056.doc [Citado el 26/02/2009].

14. Modelo de Diseño. Ministerio del Poder Popular para las Telecomunicaciones y la Informática. [En línea]. Disponible en: http://merinde.rinde.gob.ve/index.php?option=com_content&task=view&id=92&Itemid=296 [Citado el 25/03/2009].

15. MSc. CIUDAD, F. *ApEM-L como una nueva solución a la modelación de aplicaciones educativas multimedia en la UC.* Universidad de las Ciencias Informáticas (UCI), Ciudad de la Habana. 2007.

16. ORTÍN, M., García, J. Modelado basado en roles con UML. [En línea]. Disponible en: http://dis.um.es/~jmolina/rolesuml.pdf [Citado el 4/03/2009].

17. PÉREZ, Y. *Metodologías para el desarrollo de software educativo: Un estudio comparativo.* Universidad de las Ciencias Informáticas (UCI), Ciudad de la Habana, Cuba. 2006.

18. SAUER, S. y ENGELS, G. *Extending UML for Modeling of Multimedia Applications* [En línea]. Disponible en http://www.itec.uni-klu.ac.at/~harald/proseminar02/sauer1.pdf. 2001. [Citado el 5/02/2009].

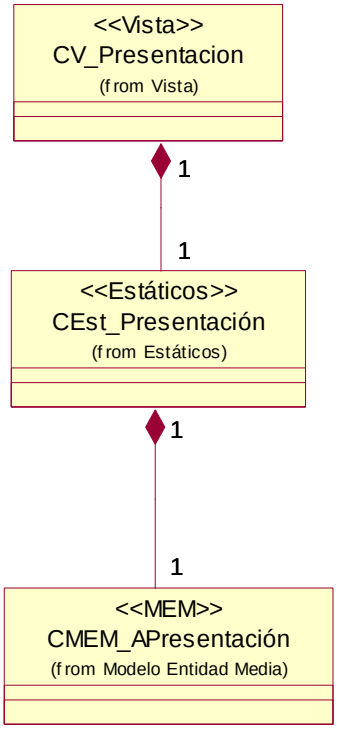
19. SAUER, S. *UML & OMMMA-L, 2001.*

20. VALVERDE, J. Universidad de Extremadura. Diseño de materiales educativos multimedia 2006 [En línea]. Disponible en: http://www.unex.es/didactica/Tecnologia_Educativa/info03J.htm [Citado el 25/02/2009].

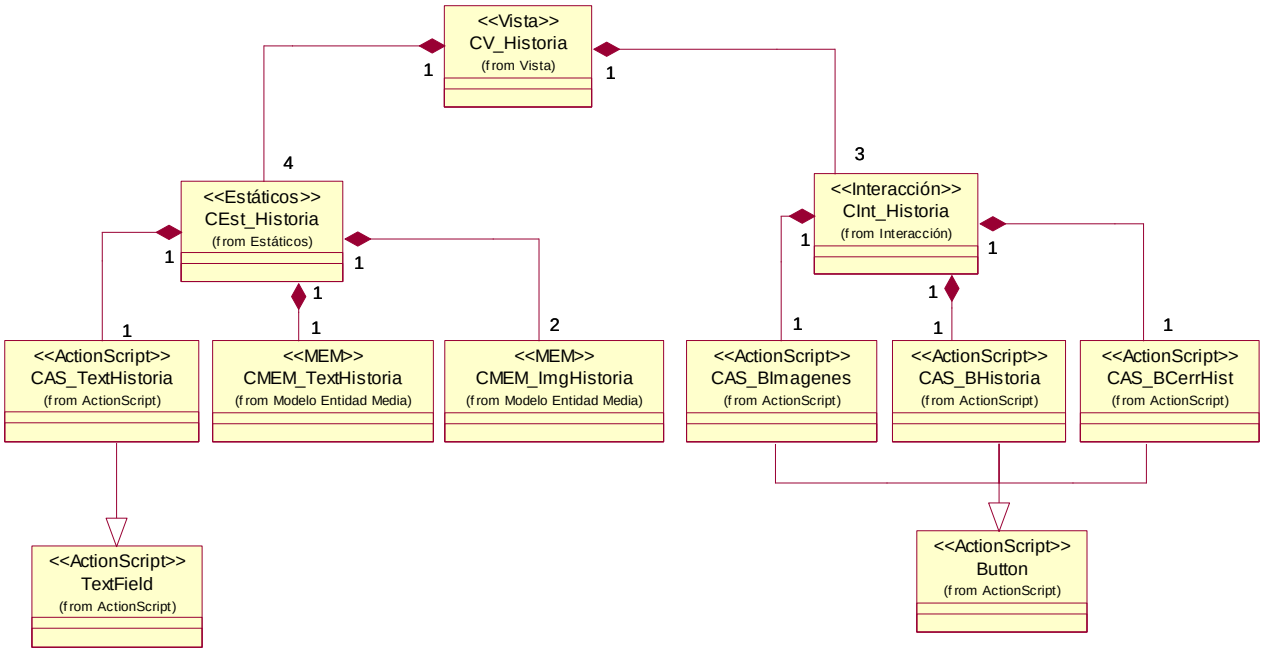
21. VALVERDE, J. Universidad de Extremadura. Diseño de materiales educativos multimedia 2006 [En línea]. Disponible en: http://www.unex.es/didactica/Tecnologia_Educativa/guion11.htm [Citado el 25/02/2009].

ANEXOS

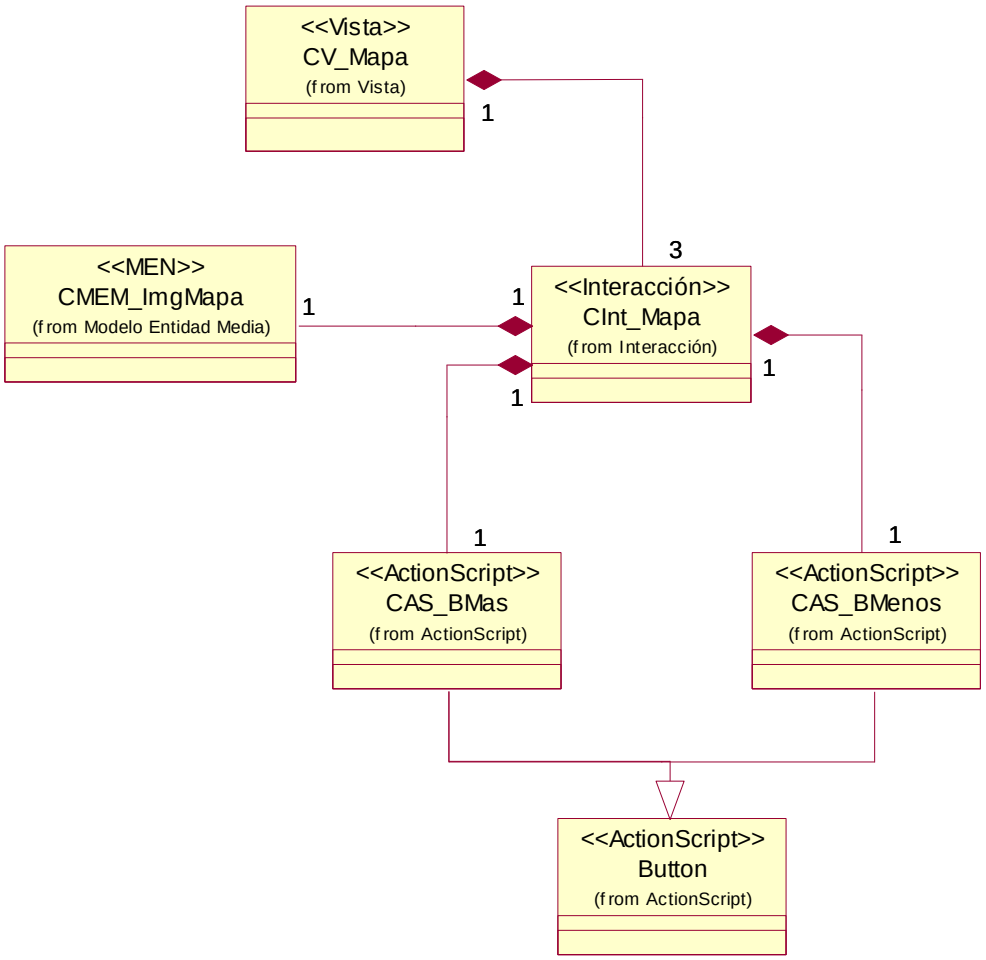
ANEXO 1: DEP Vista Presentación



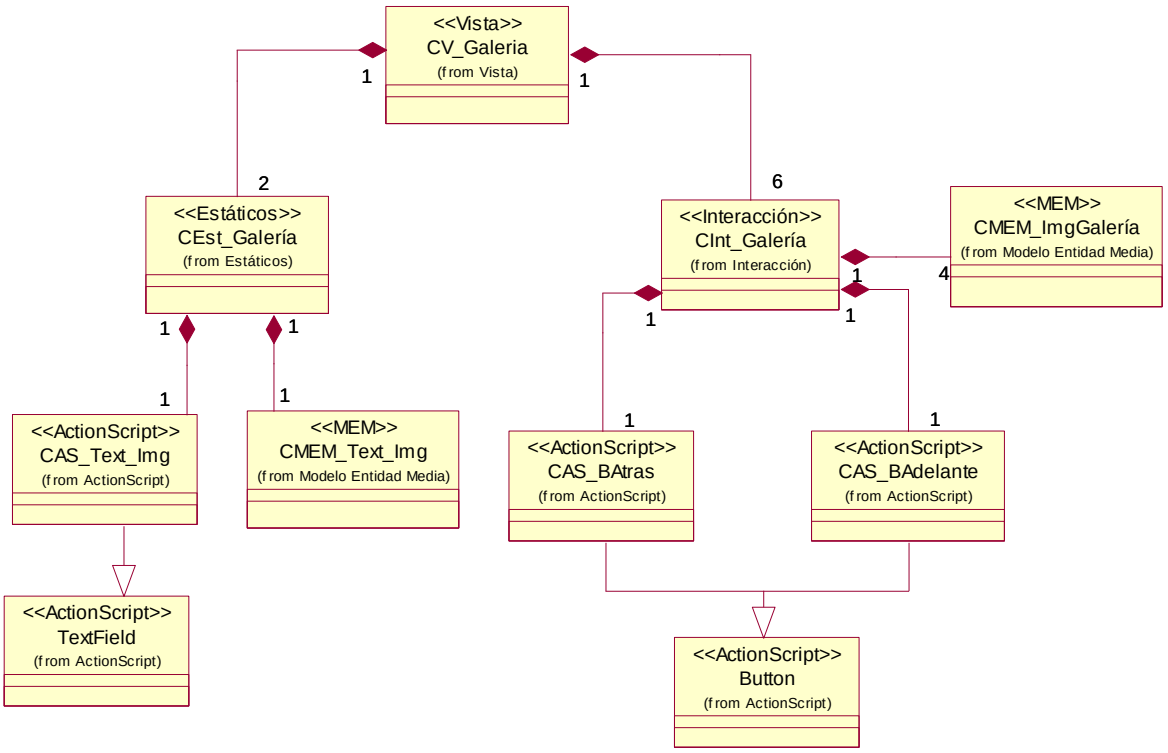
ANEXO 2: DEP Vista de Presentación Historia



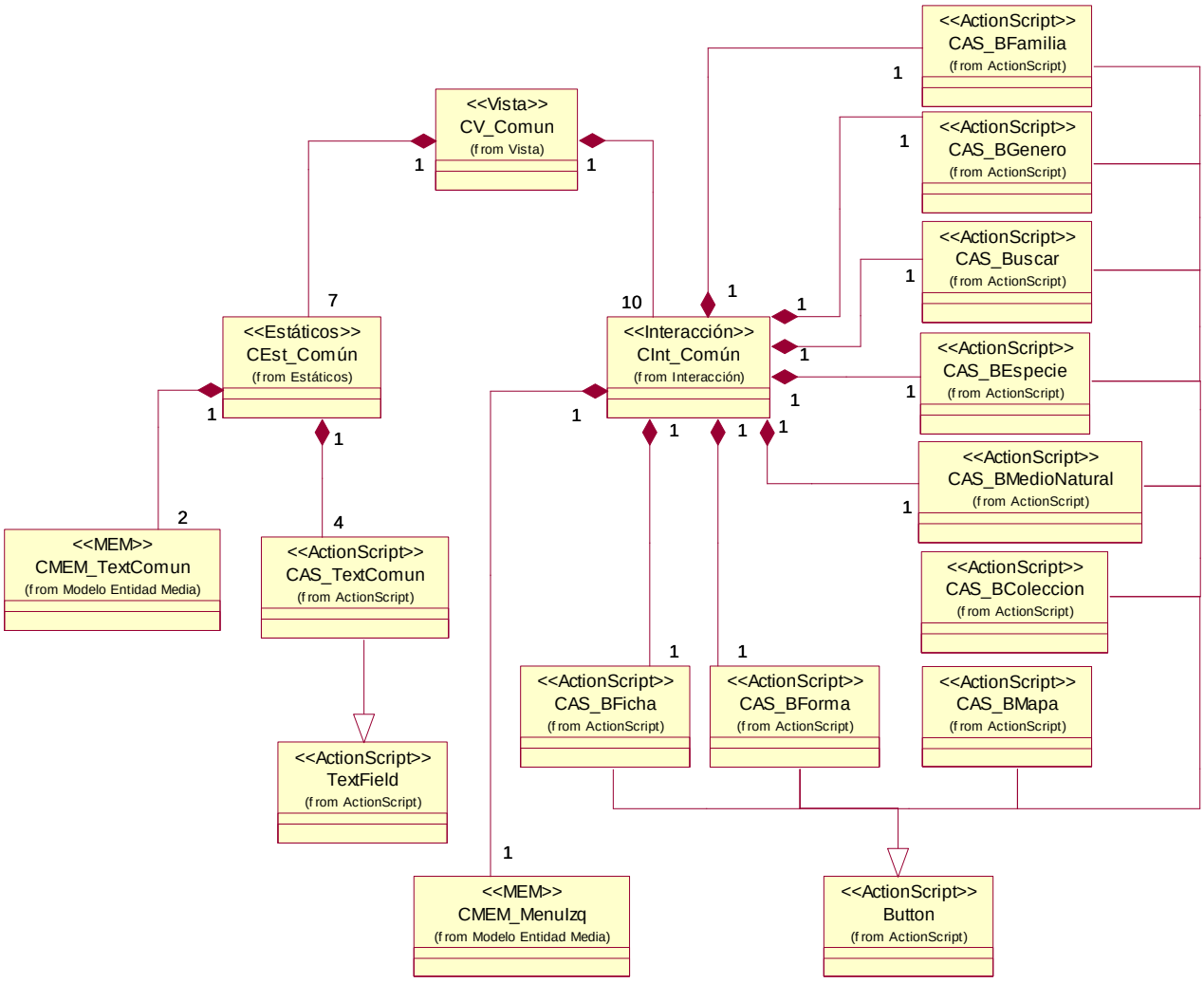
ANEXO 3: DEP Vista de Presentación Mapa



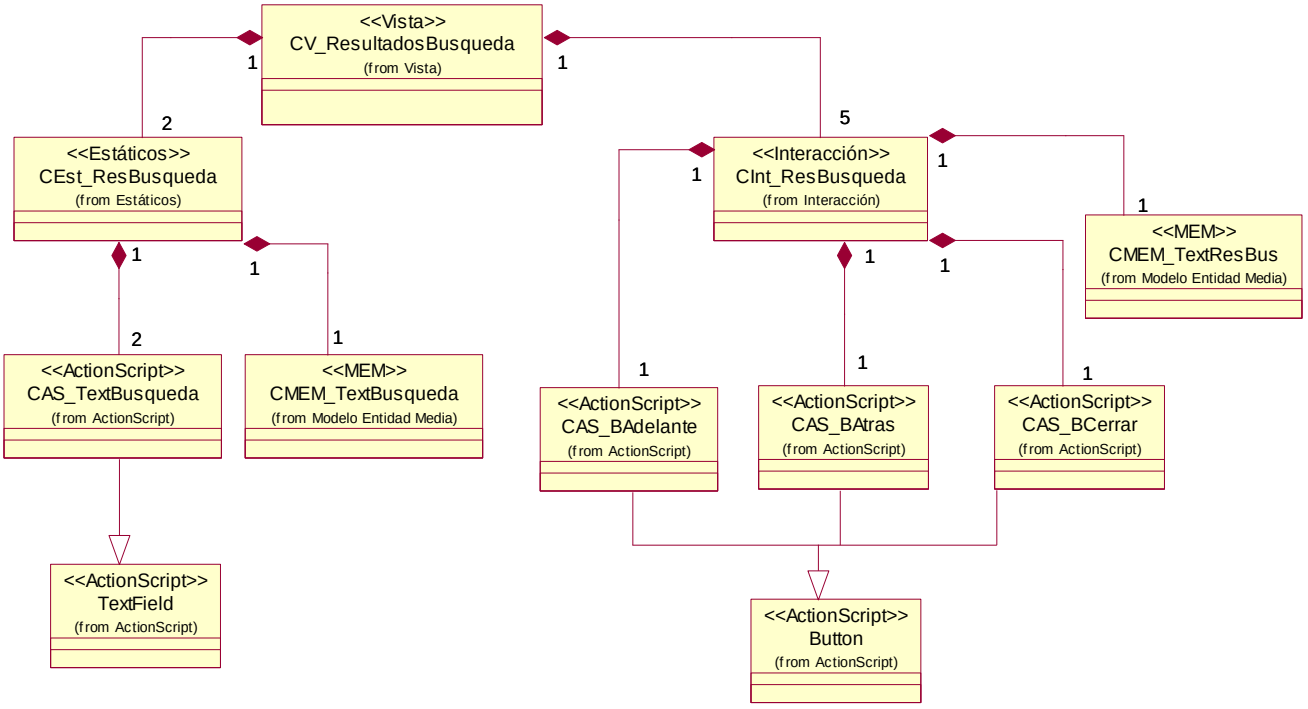
ANEXO 4: DEP Vista de Presentación Galería



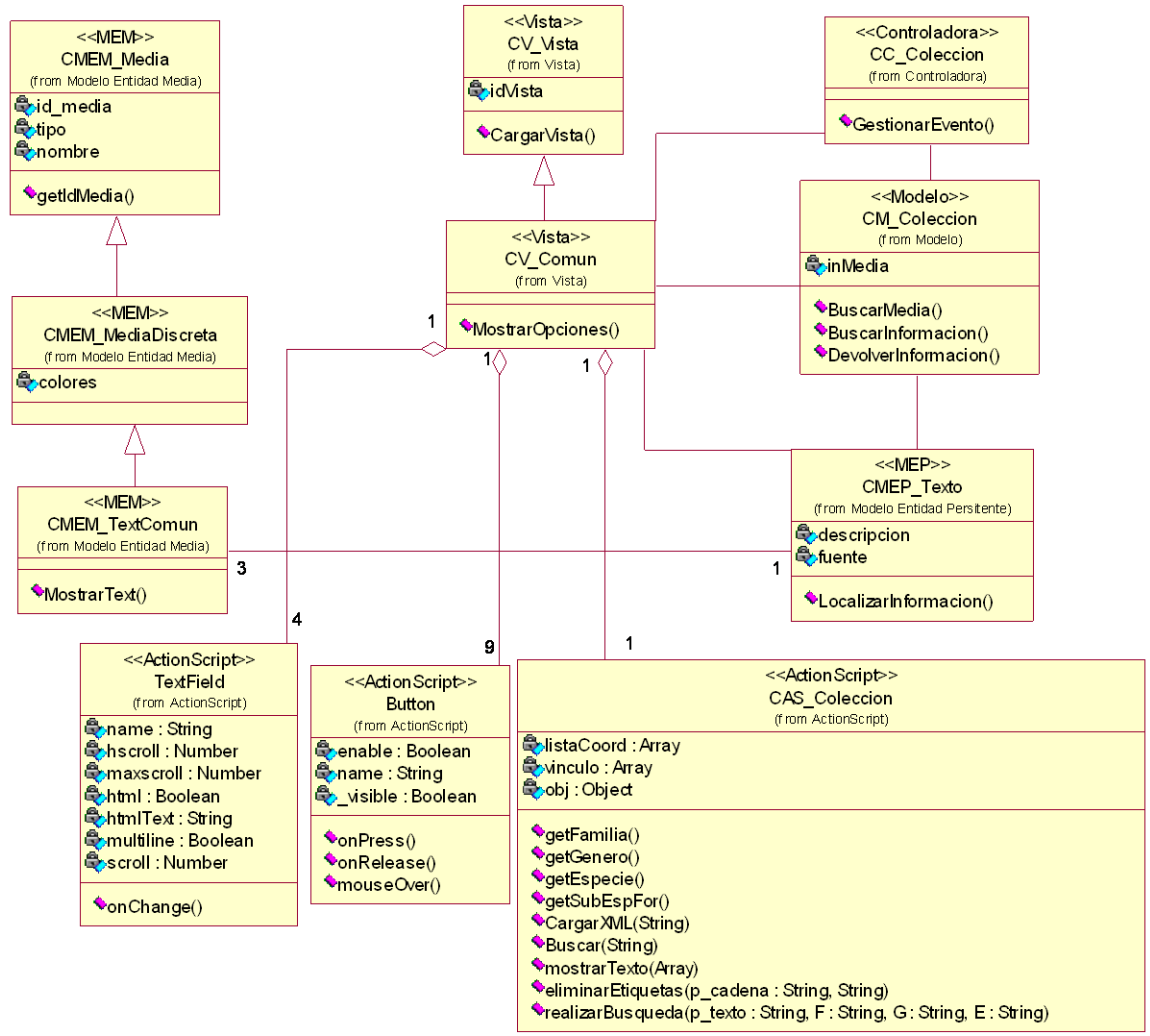
ANEXO 5: DEP Vista de Presentación Común



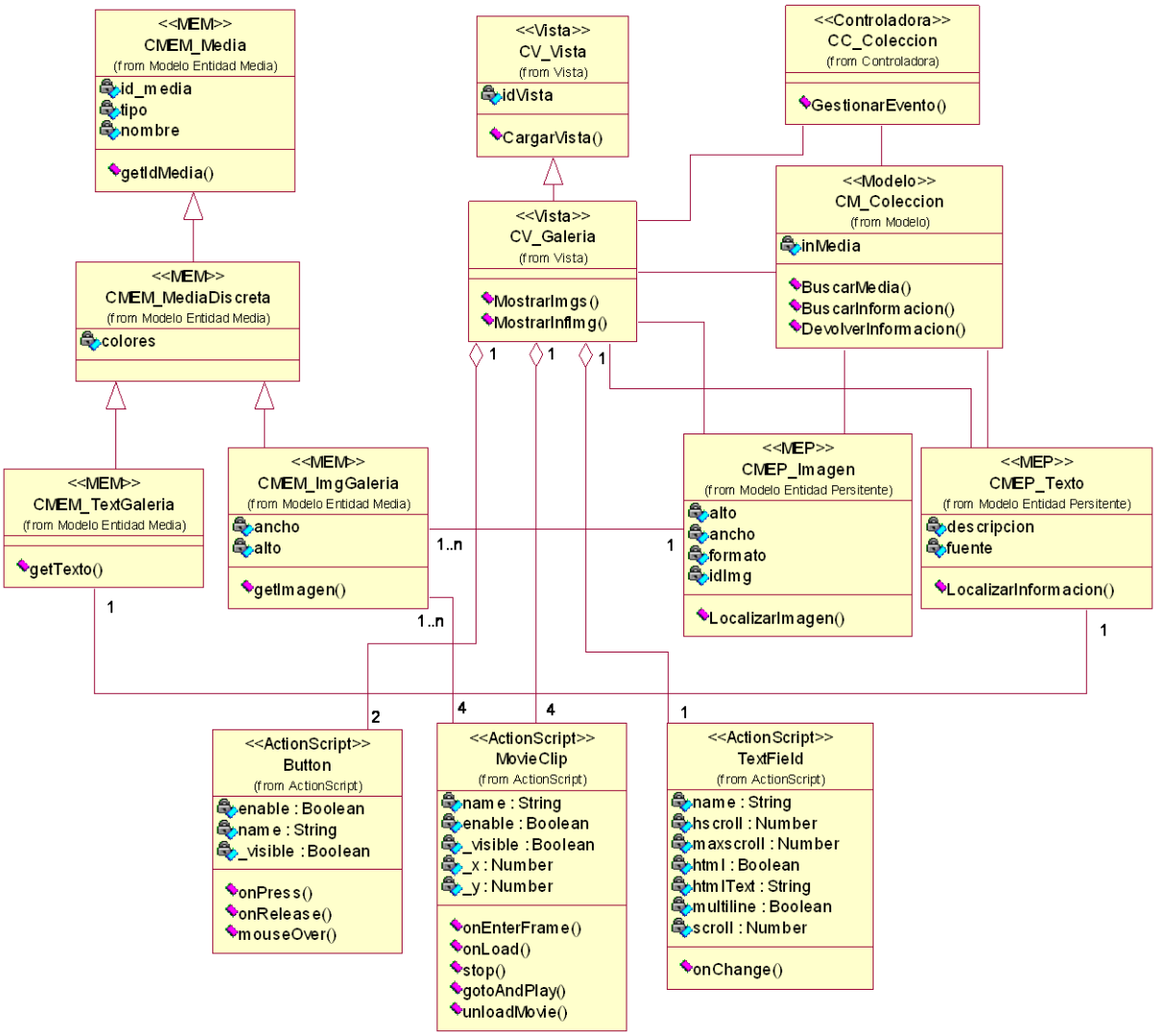
ANEXO 6: DEP Vista de Presentación ResultadosBúsqueda



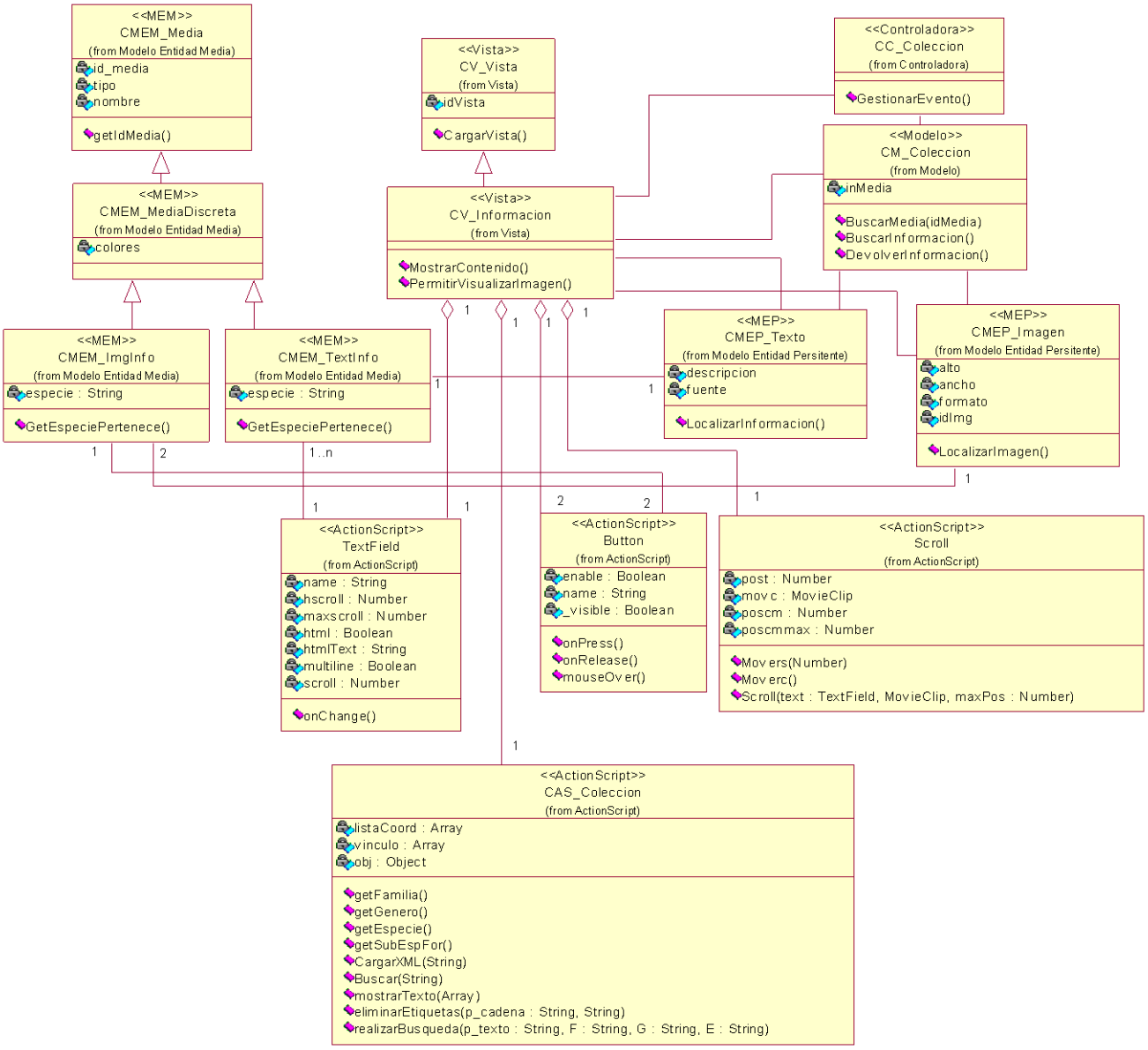
ANEXO 7: DCD Vista Común



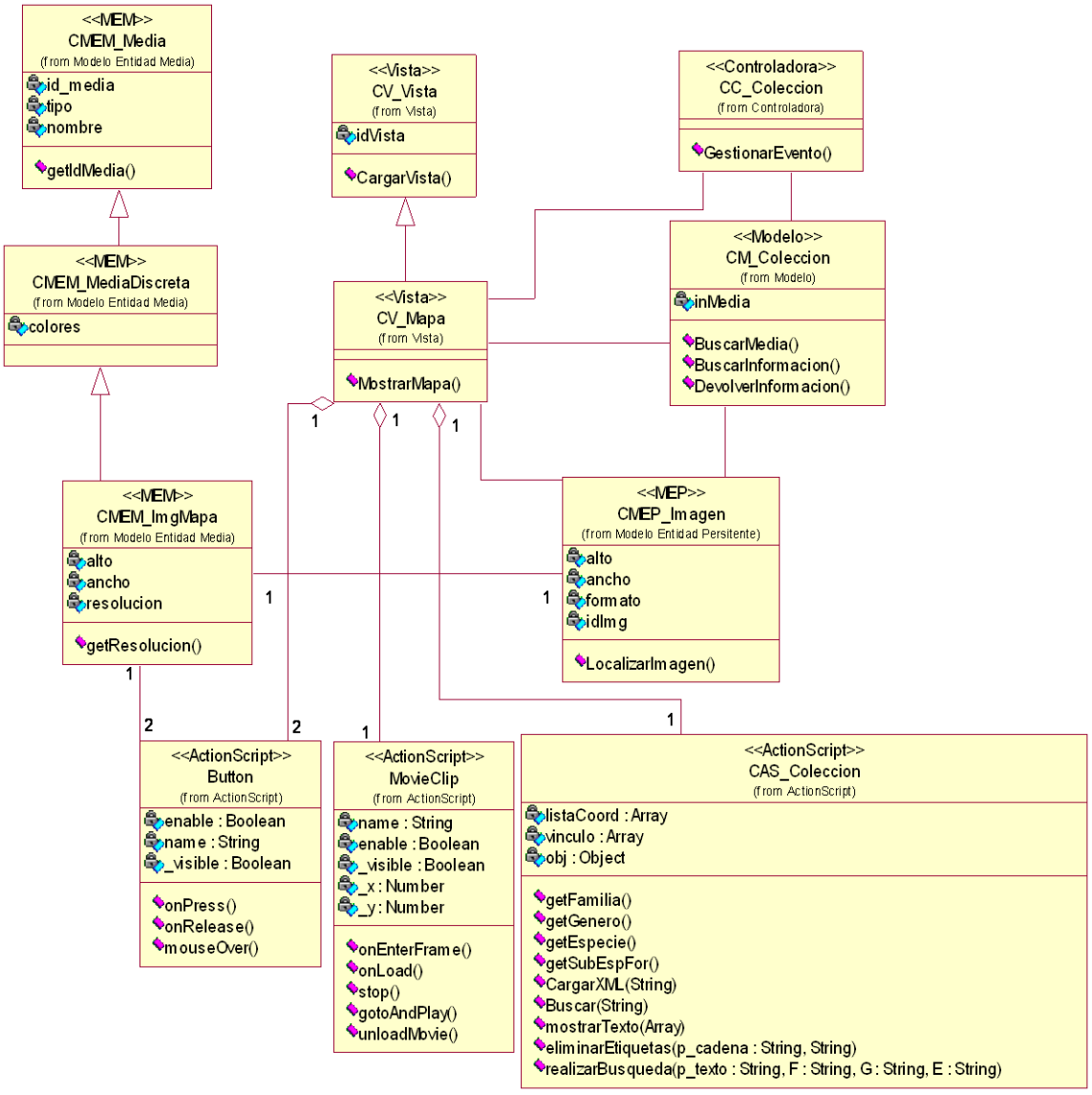
ANEXO 8: DCD Vista Galería



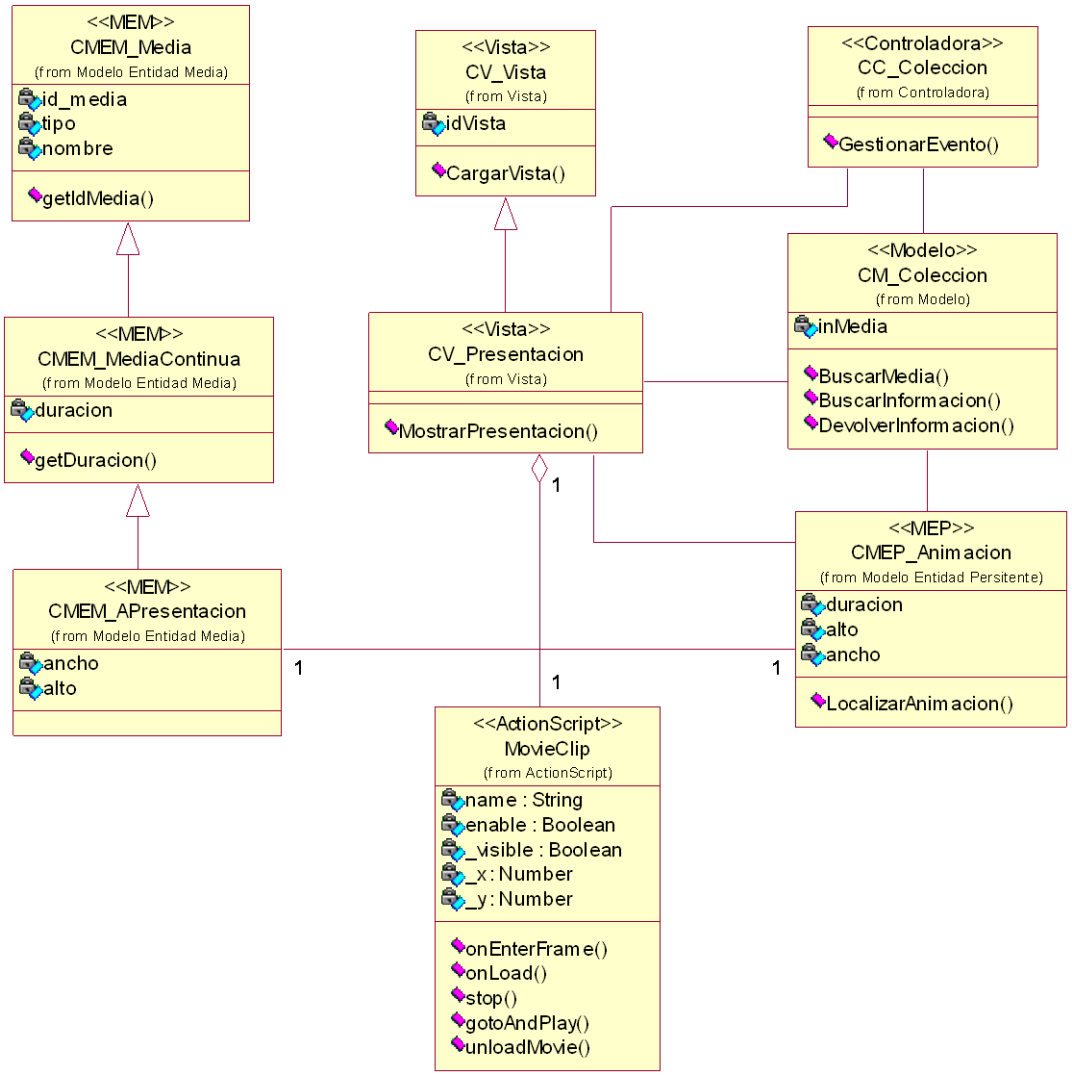
ANEXO 9: DCD Vista Información



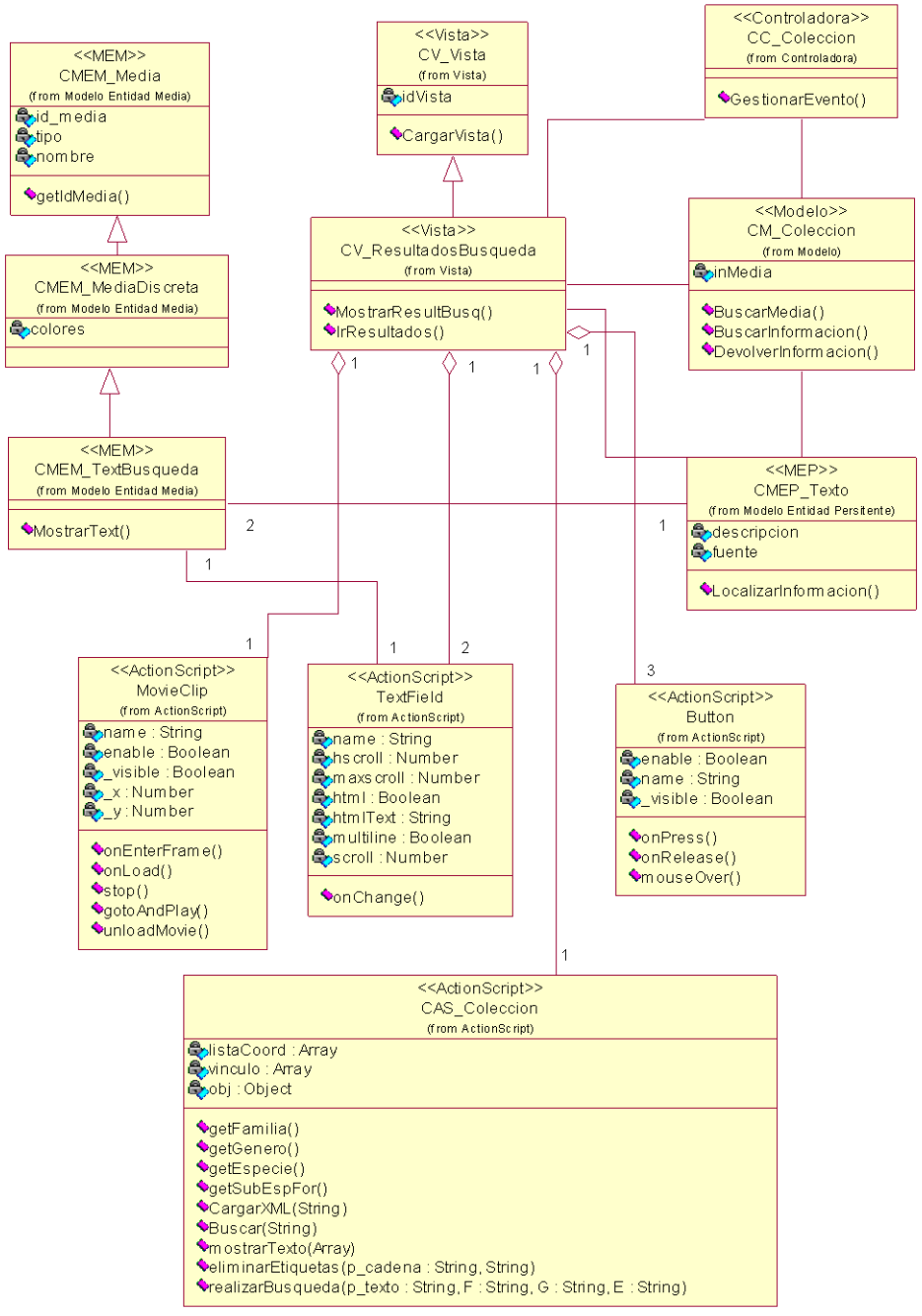
ANEXO 10: DCD Vista Mapa



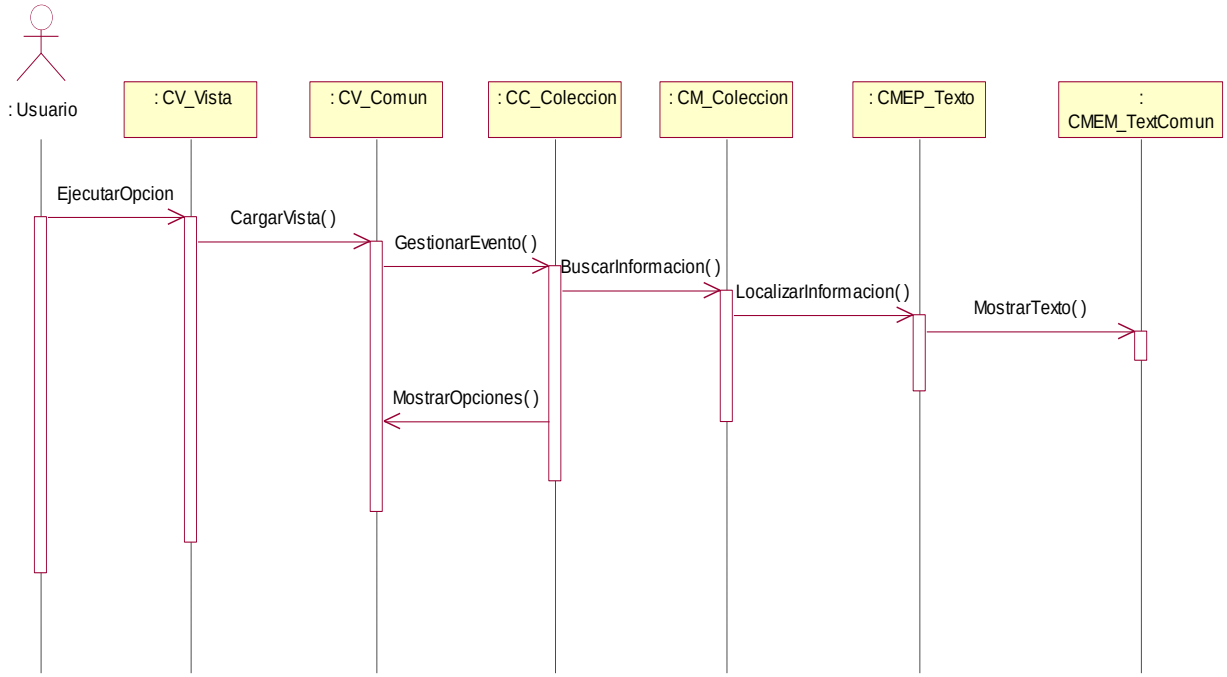
ANEXO 11: DCD Vista Presentación



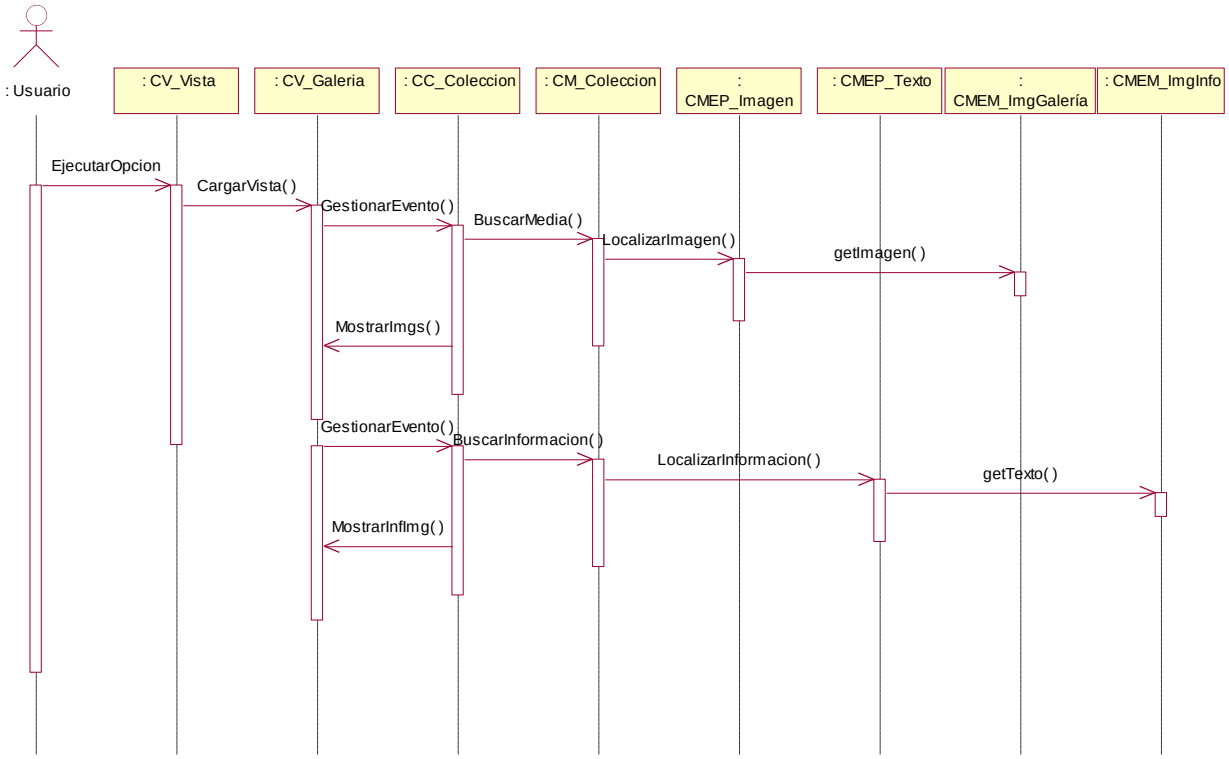
ANEXO 12: DCD Vista ResultadosBúsqueda



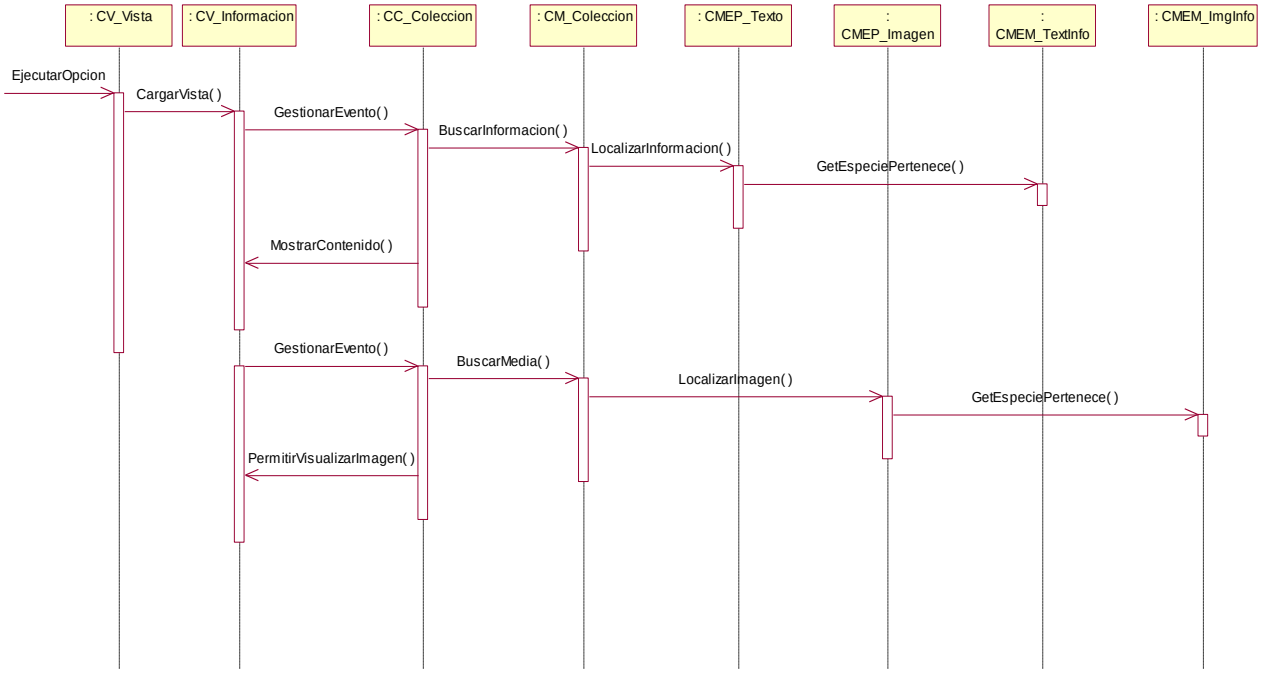
ANEXO 13: DS Vista Común



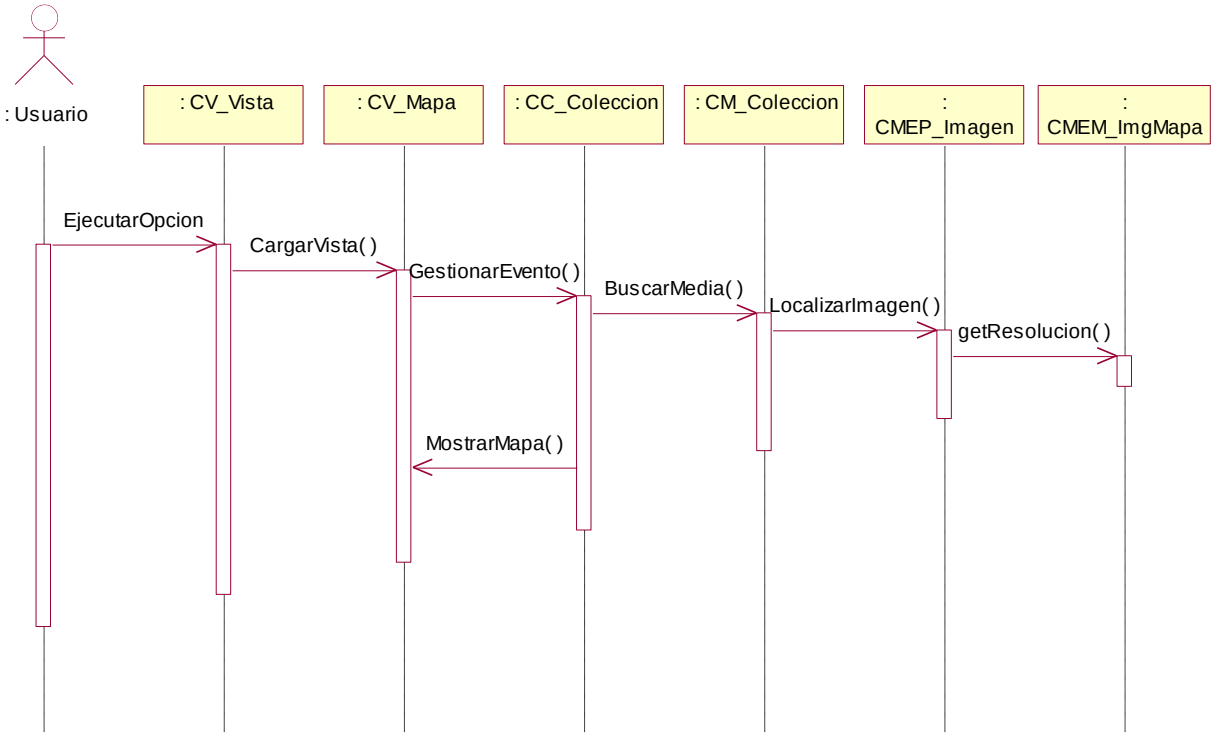
ANEXO 14: DS Vista Galería



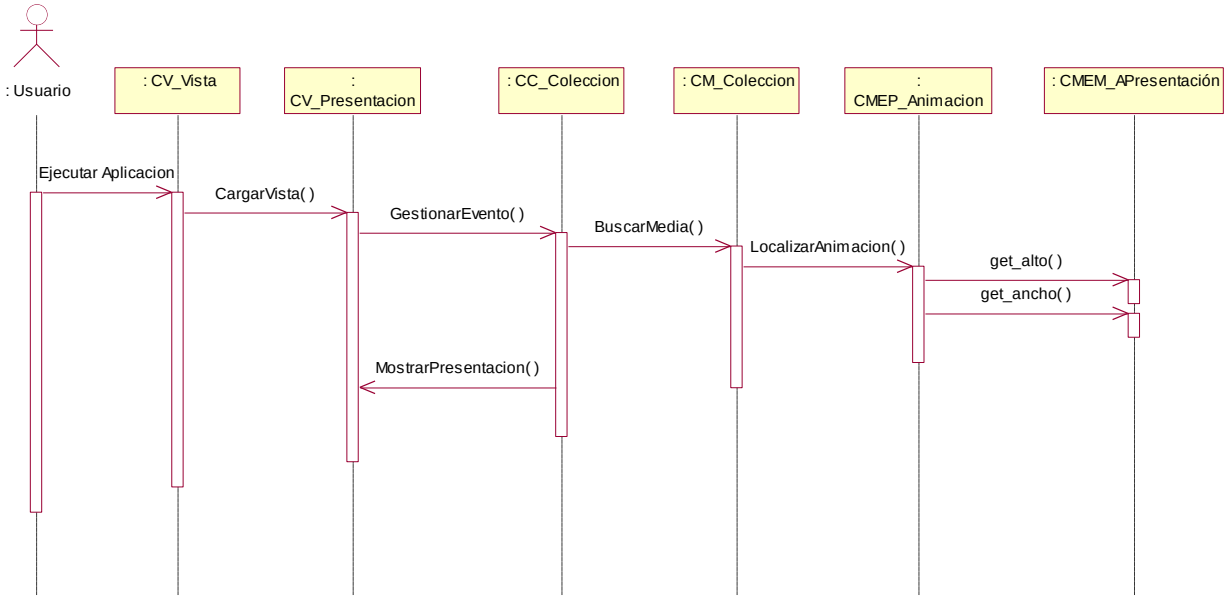
ANEXO 15: DS Vista Información



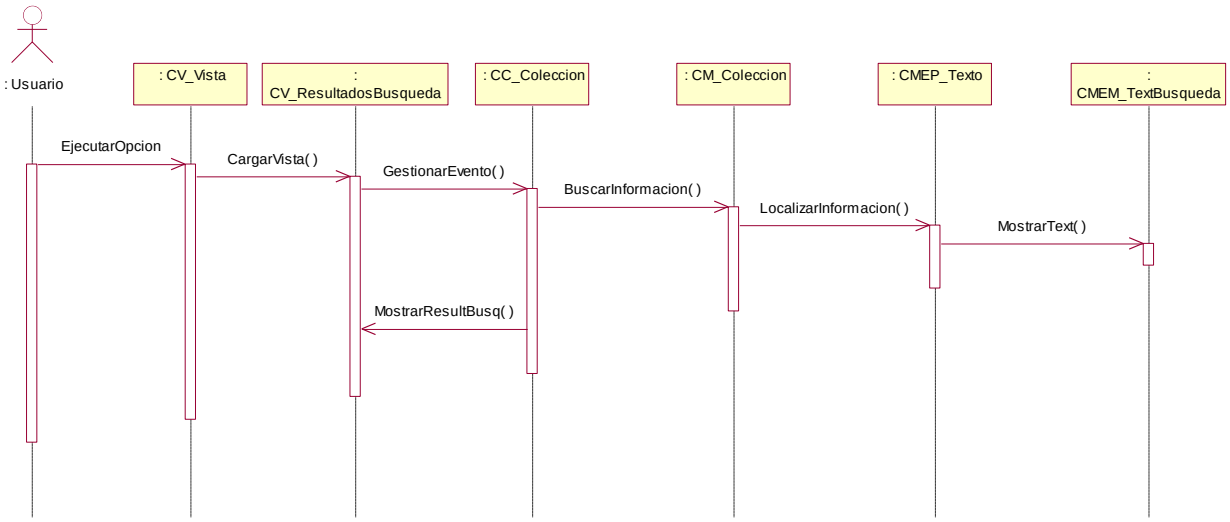
ANEXO 16: DS Vista Mapa



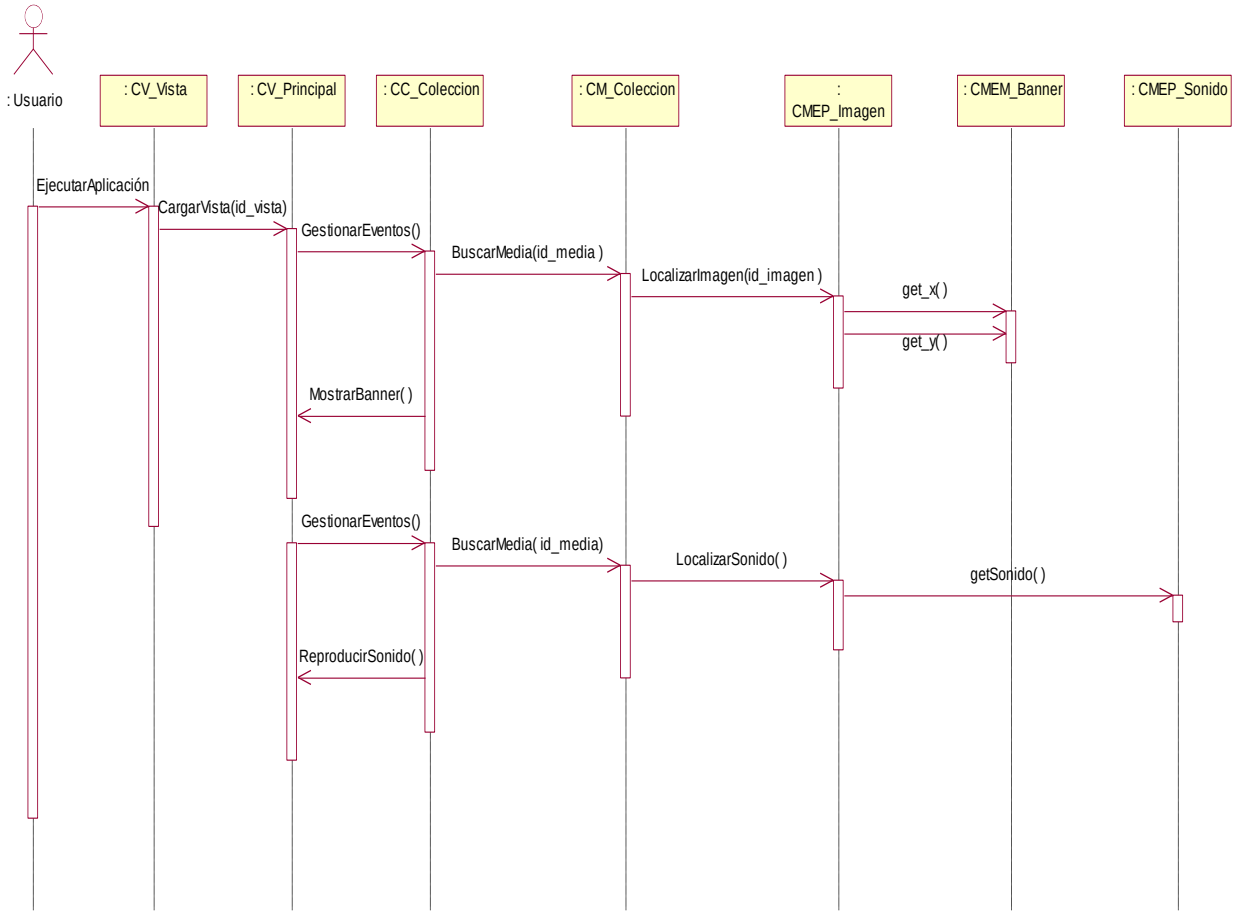
ANEXO 17: DS Vista Presentación



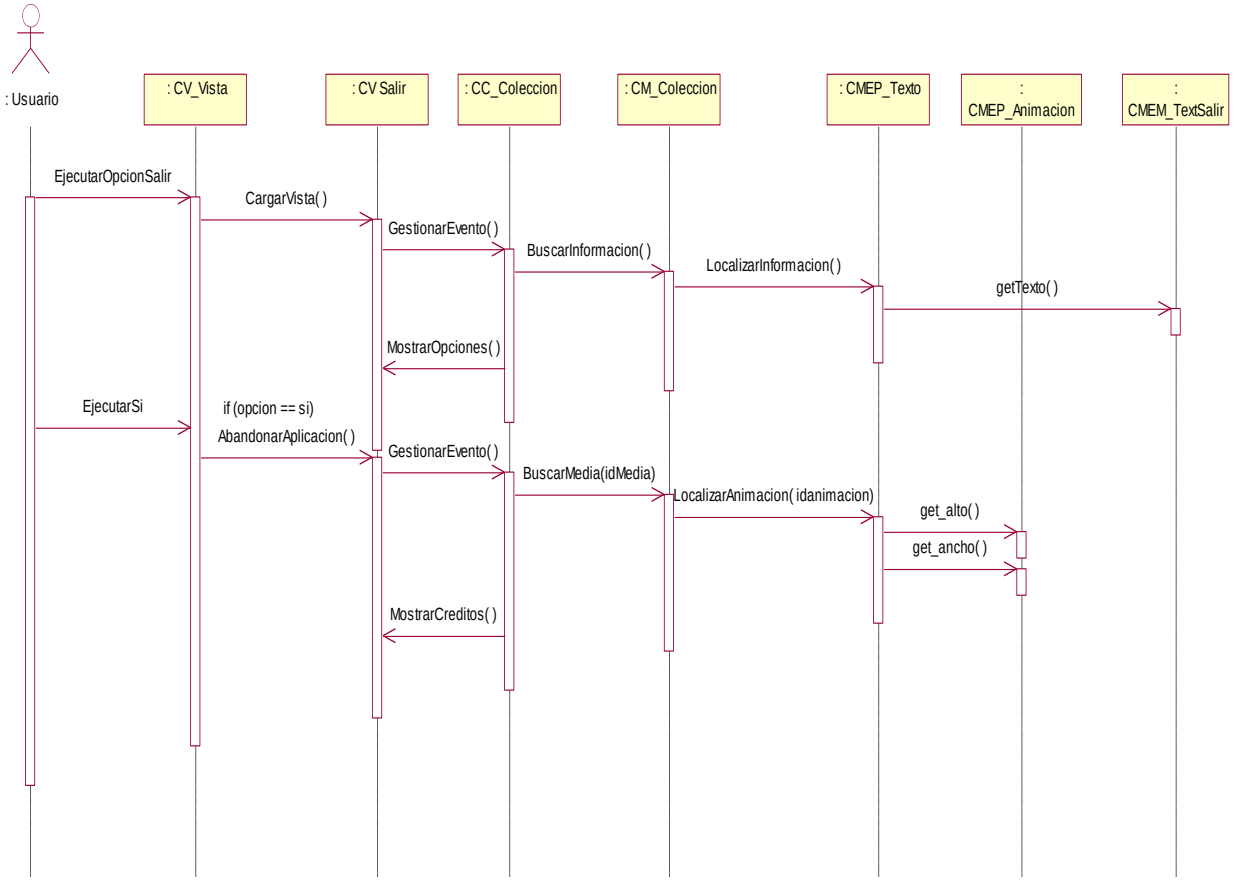
ANEXO 18: DS Vista ResultadosBúsqueda



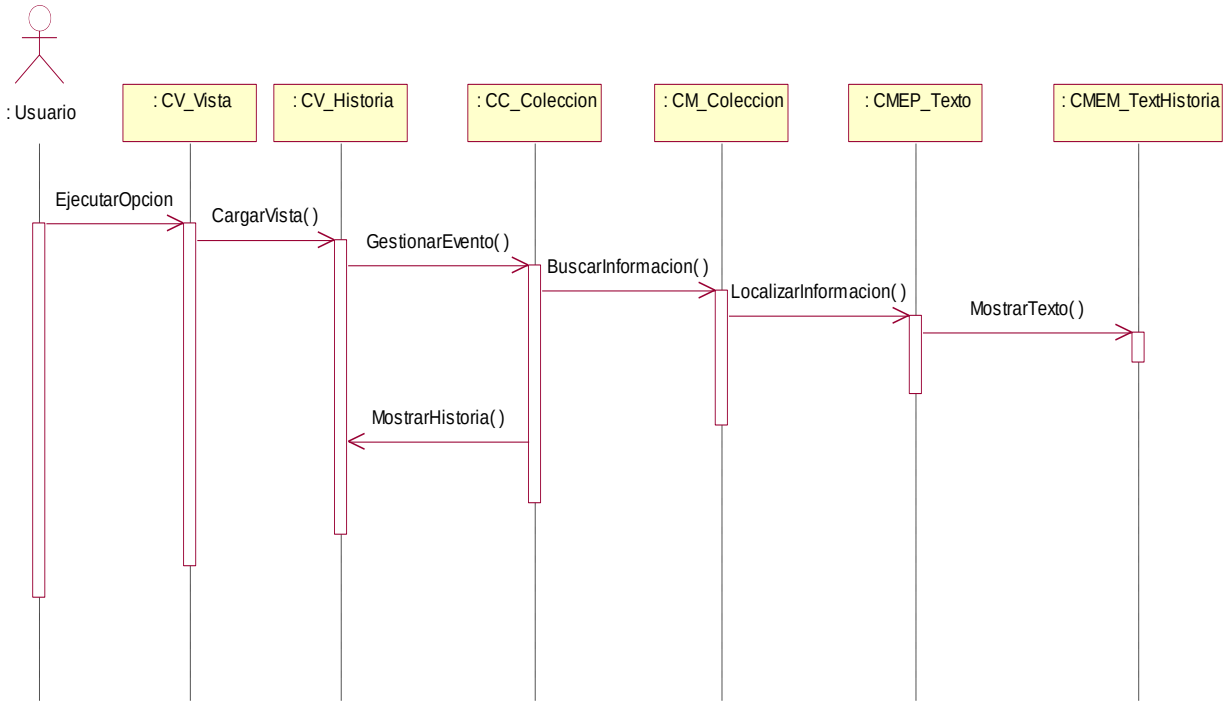
ANEXO 19: DS Vista Principal



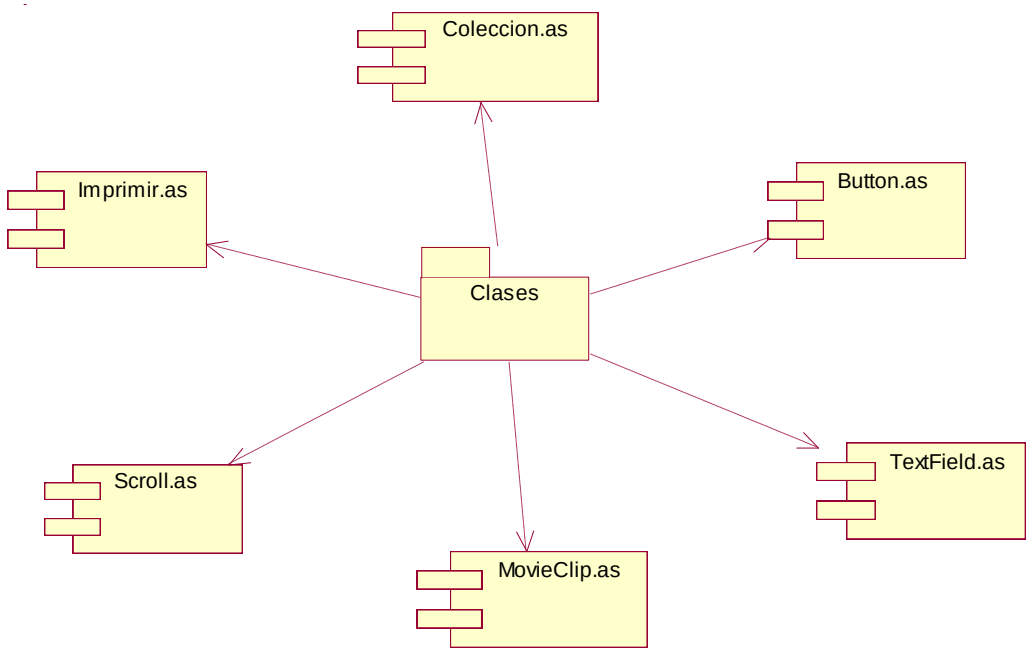
ANEXO 20: DS Vista Salir



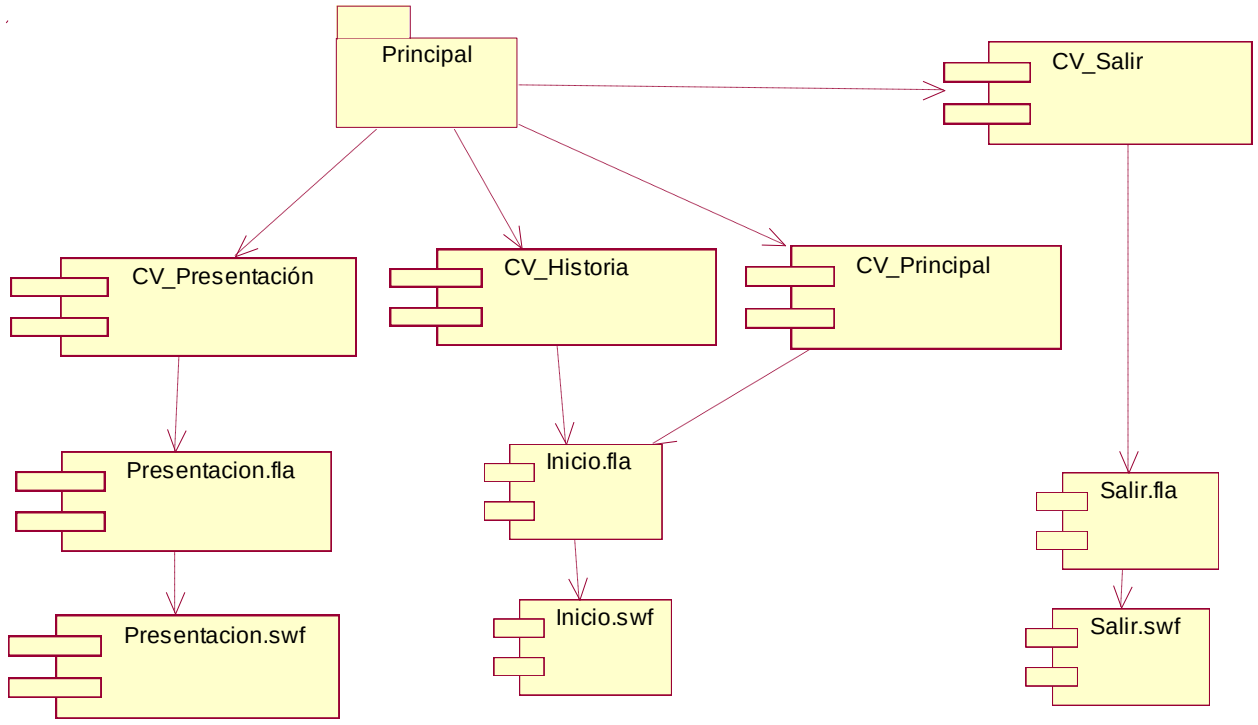
ANEXO 21: DS Vista Historia



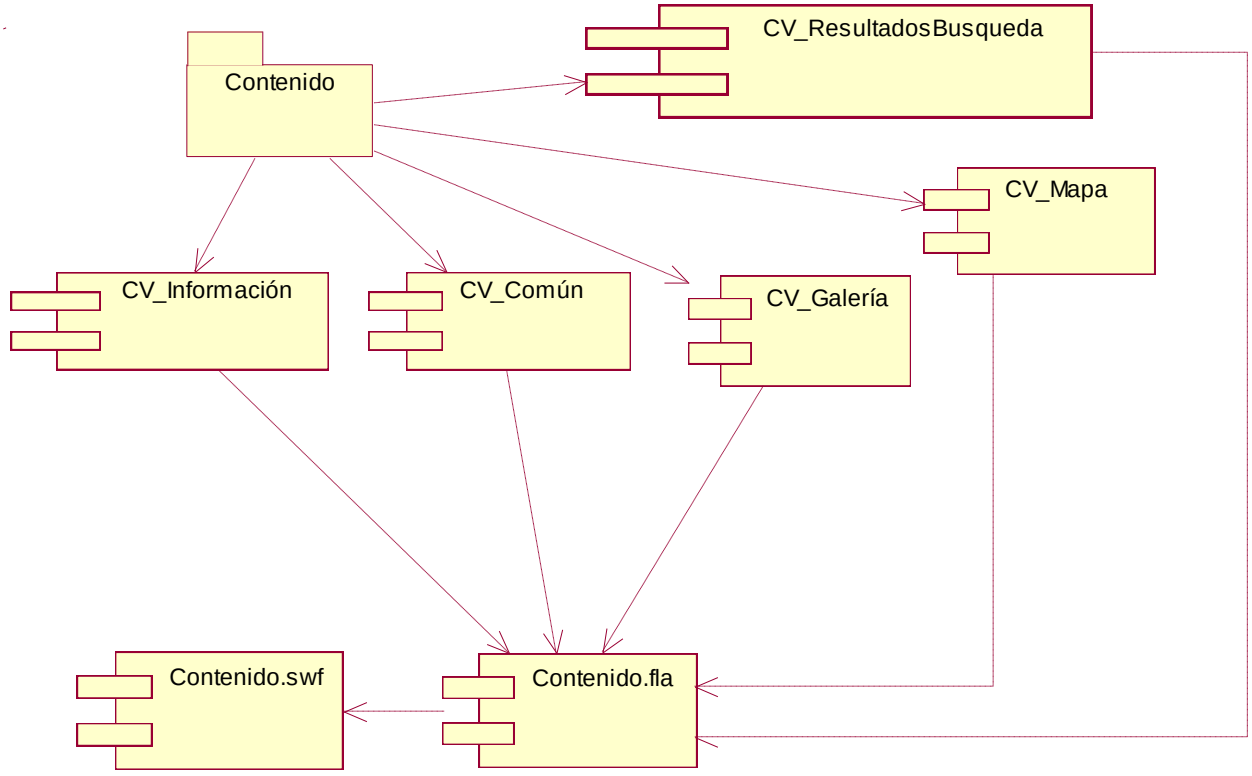
ANEXO 22: Diagrama de componentes Paquete Clases



ANEXO 23: Diagrama de componentes Paquete Principal



ANEXO 24: Diagrama de componentes Paquete Contenido



GLOSARIO DE TÉRMINOS

Apple Inc.: es una empresa multinacional estadounidense que diseña y produce equipos electrónicos y software.

Bugs: es un defecto de software es el resultado de un fallo o deficiencia durante el proceso de creación de software.

CASE: (**C**omputer **A**ided **S**oftware **E**ngineering) En su traducción al español significa Ingeniería de Software Asistida por Computación. Es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso de automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

CITMA: Ministerio de Ciencia, Tecnología y Medio Ambiente.

Componente: es parte modular de un sistema, desplegable y reemplazable que encapsula implementación y un conjunto de interfaces y proporciona la realización de los mismos. Un componente típicamente contiene clases y puede ser implementado por uno o más artefactos (ficheros ejecutables, binarios, etc.).

Diagrama: es una representación gráfica en el que se muestran las relaciones entre las diferentes partes de un conjunto o sistema.

Fotograma: en Flash es un instante o momento de una película, es equivalente a cuadro de una película. Es posible agregar, mover, eliminar, cortar, pegar y limpiar fotogramas.

IDO: Instituto de Oceanología.

Modelo: es una abstracción de un sistema del mundo real, considerando un propósito.

OCL: (**O**bject **C**onstraint **L**anguage, lenguaje de restricciones de objetos) Es un conjunto de notaciones precisas de lenguaje textual para expresar restricciones que no pueden ser expresadas en la notación de diagramas estándar usada en UML. La semántica de OCL se aplica sobre la base de la construcción de herramientas CASE que soportan comprobaciones de integridad sobre todos los modelos UML.

Pétreo: viene de piedra, se llama así de forma "vulgar" o como nombre común a los corales escleractíneos, también se les llama corales blancos.

Ráster: es una malla o matriz regular de celdas de un área determinada. Una imagen ráster o rasterizada es sinónimo de imagen matricial o bitmap.

Requerimiento: es una característica, propiedad o comportamiento que se desea para el sistema.

Sistemática: es una latinización del griego (*systema)* y se usa a partir del Systema Natura de Linneo, se aplica al sistema establecido para clasificar a los organismos de acuerdo a su historia evolutiva o filogenética.

SWF: es un formato de archivo de gráficos vectoriales creado por la empresa Macromedia. Los archivos SWF suelen ser suficientemente pequeños para ser publicados en la web en forma de animaciones con diversas funciones y grados de interactividad.

Taxonomía: (viene del griego taxis=ordenamiento y nomos=ley) establece los métodos y fines de la clasificación. Es la organización que a través de leyes y jerarquías de los grupos define los nombres en que se dividen los animales y plantas.