Universidad de las Ciencias Informáticas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

"Framework para la generación de interfaces gráficas para la plataforma educativa Dolphin".

Autores:

Leynier Viquillón Lavorí

Pedro Luis Rojas Lemus

Tutores:

Ing. Yoennis Garrido Vargas

Ing. José Antonio Soto Pérez

Ing. Jorge Antonio Díaz Gutiérrez

Ciudad de La Habana, junio del 2010.

"Año 52 de la Revolución".

"El mundo camina hacia la era electrónica... Todo indica que esta ciencia se constituirá en algo así como una medida del desarrollo; quien la domine será un país vanguardia. Vamos a volcar nuestros esfuerzos en este sentido con audacia revolucionaria."

Ernesto Che Guevara de la Serna.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores del presente trabajo y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Pedro Luis Rojas Lemus

_____

Firma del Autor

Leynier Viquillón Lavorí

_____

Firma del Autor

Jorge Antonio Díaz Gutiérrez

_____

Firma del Tutor

José Antonio Soto Pérez

_____

Firma del Tutor

Yoennis Garrido Vargas

_____

Firma del Tutor

Agradecimientos

A mi madre Cristina Claire, mis abuelos Dulce María y Roberto, mi padre Luis Manuel, mis segundos padres Eddy Manuel y Maribel, mi hermana Jaila; mis hermanos Yolexi, Osmani y Andri, por ser las personas a las que debo todo lo que ha ocurrido en mi vida. Por ellos es que he podido forjarme en la persona que realmente soy, pues me dieron la fuerza y alegría para seguir adelante y alcanzar mi meta.

A mi bella y excepcional novia Yaneisi Acosta Navarro (Tata), por su amabilidad y dedicación, por apoyarme, por creer en mí, pero sobre todo, por todo el amor que me brindaste y los momentos compartidos en estos cinco años junto a mí, haciendo la vida muy feliz.

A mis magníficos tutores Yoennis, Jorgito, Jose, por sus sugerencias para la elaboración de la presente tesis y por el apoyo brindado a lo largo de todo este año, por la confianza depositada en mí, los consejos constante y la dedicación absoluta.

A Nilber, quien aunque no fue mi tutor, me brindó su apoyo para el desarrollo de mi trabajo.

A mis compañeros de aula: Pedro (pedruk0) mi compañero de tesis y persona con la que he resuelto el tramo final de este trabajo. Héctor (Curro o Teto), Alexander (Alex), Franklin (Bambino), Leonardo (Leo), Luis Felipe (Felipao), Elvin Lara, Aldo, Yunier (Lukas), Yolanda, Dayana, Yadira, Meylin, Yailet, Karen con quienes he compartido estos 5 años de vida universitaria y me han brindado su apoyo en este trabajo durante el último año. Les debo un entorno lleno de generosidad y de talento, que llega mucho más allá del esfuerzo que absorbe el trabajo diario. Gracias a todos aunque sea una pequeña manera de demostrarle que en verdad valió la pena conocerles y agradecerles el haberme brindado todo el apoyo, colaboración, ánimo y sobre todo cariño y amistad. Los quiero mucho y nunca los olvidaré

A la Universidad de las Ciencias Informáticas, por su apoyo para completar mi formación profesional para poder ser una persona útil de esta sociedad. Y en general, a todos los integrantes de la Facultad 8. En particular a mi decano y amigo Pedro Luis Basulto Ramírez.

Un millón de palabras o lágrimas, no pueden agradecerles a todos. Cultivar verdaderos amigos requiere dedicación y tiempo. No puedo escribir otra cosa que gracias, y justamente, agradecer a todos por existir, por brindarme sus segundos, por aguantarme con paciencia, por ser capaces de apoyarme, por callar, por hablar, por ser mis amigos.

Leynier

Agradecimientos

A mi mamá, gracias por siempre estar presente, por ser mi principal fuente de inspiración para avanzar en la vida, por el esfuerzo constante y por una vida de sacrificios, te quiero, eres la mejor del mundo, te regalo mi título.

A mis hermanas Dania y Anita, gracias por mantenernos siempre unidos en los buenos y malos momentos.

A mis abuelos, gracias por los tantos consejos que me sirvieron de apoyo en los momentos más difíciles.

A mi familia, gracias por darme el apoyo y la comprensión necesaria para cumplir con mi sueño de convertirme en ingeniero.

A mi tutor Yoenni, gracias por su comprensión, esfuerzo, apoyo y ayuda incondicional.

A Jorge, José y Nilber, gracias por siempre estar presentes cuando teníamos una duda.

A mis compañeros de grupo y amigos, gracias por compartir juntos momentos de preocupación y alegría, en especial al Viqui, Leo, Hector, Franklin, Alex, Lucas, Felipe y para mi querida amiga Yudis, no piense que me he olvidado de ti.

A todas las personas que se preocuparon por el desarrollo de esta tesis.

A la Revolución y nuestro Comandante Fidel por darme la posibilidad de realizar mis sueños.

A todos gracias y les regalo la satisfacción de que ya soy Ingeniero.

Pedro Luis

A mi madre, porque todo lo que he logrado en la vida es gracias a su amor y dedicación.

A mi abuelito Roberto, que no pudo verme graduado.

A mi abuelita Dulce María, que tanto lucha por mi bienestar.

A mi familia completa, porque este sueño es parte de ellos.

Leynier

A mi madre Eloida por ser tan especial y darme todo en esta vida.

A toda mi familia por darme todo su apoyo siempre que lo he necesitado.

Pedro Luis

Resumen

La evolución de las plataformas educativas se encuentra estrechamente ligada al desarrollo de la sociedad de la información y del conocimiento, y específicamente, al perfeccionamiento de los sistemas educativos, que tienden de manera incesante a adaptarse a las necesidades reales del mundo laboral.

En la Universidad de las Ciencias Informáticas se han desarrollados productos que contribuyen al perfeccionamiento de los métodos de enseñanza-aprendizaje, uno de estos software es “Español para no hispano hablantes”, el cual no permite gestionar los contenidos que en él se exponen, ni la construcción de nuevas interfaces gráficas, que posibiliten combinar las medias necesarias de un curso. Para dar solución a los problemas mencionados, se realizó la investigación que lleva como título "Framework para la generación de interfaces gráficas para la plataforma educativa Dolphin", con el objetivo de desarrollar un framework que permita generar automáticamente las interfaces gráficas de los contenidos de la plataforma educativa Dolphin, mediante el trabajo con archivos XML.

Para el logro del objetivo propuesto se realizó una amplia revisión bibliográfica, que permitió elaborar y exponer la fundamentación teórica de los frameworks, así como las soluciones similares y las herramientas y tecnologías utilizadas. Efectuando el análisis, diseño e implementación del framework, guiado por la metodología de desarrollo XP, modelando sus clases con UML y utilizando como lenguaje de programación ActionScript 3.0. La construcción del sistema propuesto tiene un carácter iterativo, que permite la inserción de nuevas funcionalidades y la rectificación de errores, que posibilitan la reutilización de código y componentes en proyectos similares.

Palabras claves: XML, ActionScript 3.0, Interfaces gráficas, Framework, Plataforma.

Introducción

El vertiginoso avance de las Tecnologías de la Información y las Comunicaciones (TICs) y su influencia en todos los ámbitos de la sociedad, han permitido no solo el incremento en los resultados de la ciencia, la producción y los servicios, sino también que se reflejan en la forma de actuar y pensar de los individuos, abriendo las puertas a una era totalmente revolucionaria, que a pesar de basarse en un concepto tan antiguo como el hombre, se encuentra en constante desarrollo y constituye un gigantesco paso de avance para la informática, provocando cambios significativos en todas las esferas, y en especial en la rama de la educación.

En la actualidad los sistemas educativos de todo el mundo se enfrentan al desafío de utilizar las tecnologías, para proveer a sus alumnos un aprendizaje más interactivo, utilizando las herramientas y conocimientos necesarios; es por ello que han surgido transformaciones en el proceso de enseñanza-aprendizaje y la forma en que docentes y alumnos acceden al conocimiento y la información.

En Cuba el sector de la educación no ha estado exento de dichos adelantos tecnológicos, por lo que ha puesto en práctica el uso y perfeccionamiento de las TICs para desarrollar software educativo fundamentalmente en formato multimedia, como complemento al aprendizaje de los alumnos con la introducción de la computadora en las aulas, permitiendo de esta forma impulsar la formación docente educativa y garantizar la eficacia de los métodos enseñanza-aprendizaje utilizados.

La Universidad de las Ciencias Informáticas (UCI), cuenta con varias facultades enfocadas a la producción de software para uso nacional y de exportación. Dentro de ellas se encuentra la Facultad 8, que asume el protagonismo en la producción de multimedia y software educativo, con el objetivo de apoyar el proceso docente educativo y facilitar el aprendizaje de los contenidos de forma dinámica mediante la combinación de textos, imágenes, videos, sonidos, juegos, actividades interactivas u otra forma de controlar los conocimientos adquiridos de cada estudiante sin aplicar un examen.

Uno de los productos que ha desarrollado la facultad 8 es “Español para no hispano hablantes”, el cual, en su primera versión no satisface las necesidades principales del cliente, pues no brinda la posibilidad de gestionar los contenidos que en él se exponen, ni permite construir nuevas interfaces gráficas que combinen textos, imágenes, videos, sonidos, actividades interactivas u otra forma de controlar los conocimientos adquiridos de cada estudiante, sin la necesidad de aplicarles un examen. Además no cuenta con un conjunto de plantillas o contenidos prediseñados que sirvan de apoyo para el diseño de los recursos didácticos y que ayuden al profesor a crear, modificar y editar los materiales que conforman el

curso, sin la necesidad de poseer conocimientos previos de programación y sin tener que contactar con el personal involucrado en el desarrollo del producto.

Debido a la situación expuesta anteriormente se plantea el siguiente **problema de investigación:** ¿Cómo generar automáticamente las interfaces gráficas de los contenidos para la plataforma educativa Dolphin?

A partir del problema de investigación se enmarca como **objeto de estudio** el proceso de desarrollo de frameworks para plataformas educativas.

El **objetivo general** de la investigación está dirigido a desarrollar un framework que posibilite generar automáticamente las interfaces gráficas de los contenidos que conforman los cursos educativos de la plataforma Dolphin, mediante el trabajo con archivos XMLs.

Se delimita como **campo de acción** el proceso de desarrollo de un framework para generar interfaces gráficas de los contenidos para la plataforma educativa Dolphin.

Se plantean los siguientes **objetivos específicos** para lograr el cumplimiento del objetivo general:

- ✓ Analizar toda la información inherente al desarrollo de APIs (Application Programming Interface) que favorezcan al software educativo.
- ✓ Realizar el diseño de clases del framework que permita generar las interfaces gráficas de los contenidos que conforman los cursos educativos de la plataforma Dolphin.
- ✓ Definir un modelo XML para cada una de las medias que conforman los contenidos de los cursos educativos de la plataforma Dolphin.
- ✓ Realizar la implementación de clases del framework que permitan generar las interfaces gráficas de los contenidos que conforman los cursos educativos de la plataforma Dolphin.

Para encaminar la investigación con vista a resolver el problema planteado se propone la siguiente **idea a defender:** el desarrollo de un framework que genere las interfaces gráficas de los contenidos que conforman los cursos educativos, permitirá definir estándares XML para cada una de las medias que conforman los cursos y cargar de forma dinámica sus contenidos, así como la reutilización de códigos y componentes en proyectos similares.

Para el desarrollo del framework se cuenta con una serie de **tareas** que ayudarán a cumplimentar su construcción:

1. Realizar una revisión bibliográfica para fundamentar la investigación.
2. Investigar y seleccionar la metodología de desarrollo para guiar el proceso de elaboración del framework.
3. Investigar y seleccionar las herramientas y tecnologías a utilizar.
4. Definir los patrones de arquitectura y estándares de diseño a utilizar.
5. Realizar un estándar XML para cada una de las medias que conforman los contenidos de los cursos educativos de la plataforma Dolphin.
6. Realizar un estándar XML para las tipologías de ejercicios de selección que conforman los cursos educativos de la plataforma Dolphin.
7. Implementar y documentar las clases del framework que gestione los contenidos que conforman los cursos educativos de la plataforma Dolphin.
8. Realizar la implementación de las plantillas o contenidos prediseñados, que sirvan de apoyo para el diseño de los recursos didácticos de la plataforma Dolphin.
9. Realizar el documento de tesis.

Fundamentación Teórica

1.1. Introducción

En la actualidad la complejidad de los sistemas de software junto con los exigentes tiempos de desarrollo manifiesta cada vez más la necesidad de eficiencia y aprovechamiento del tiempo en cada una de las tareas que se realizan durante el ciclo de vida del software. Es así como han surgido en el ámbito computacional asistentes que ayudan a optimizar tareas y a generar código automáticamente.

En este capítulo se provee una visión general de los principales conceptos relacionados a los frameworks y al proceso para su desarrollo, se centrará en la investigación de otras tecnologías y la existencia de soluciones similares que faciliten la implementación del framework. Se hace un estudio profundo del estado del arte del tema que conduce al desarrollo del presente trabajo.

1.2. Interfaz gráfica de usuario

La interfaz gráfica de usuario (en inglés Graphical User Interface, GUI) es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.

Antecedentes

En el contexto del proceso de interacción persona-ordenador, la interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico.

Las interfaces gráficas surgen de la necesidad de hacer los ordenadores más accesibles para el uso de los usuarios comunes, y permitieron que las personas pudieran acceder a un ordenador sin tener que pasar por el engorroso proceso de tener que aprender a manejar un entorno bajo línea de comandos.

Evolución

Las interfaces siguieron y siguen evolucionando; un sistema operativo de ventanas ya es algo común y corriente para la mayoría de las personas. Por otra parte el desarrollo acelerado de las Telecomunicaciones y el Internet, junto con nuevos y distintos dispositivos electrónicos como

computadoras de mano (Personal Digital Assistant o PDA, o más conocidas como Palms) y teléfonos celulares, han dado lugar a nuevas plataformas e interfaces (1).

1.3. Análisis de otras soluciones existentes

Existen varios trabajos relacionados con la generación automática de interfaces gráficas. Muchos de ellos consisten en frameworks de código libre. La mayoría de estos trabajos, incluyendo el presente, coinciden en las ventajas de partir de un lenguaje más simple y natural para definir las interfaces, y a partir de éste poder generar código para una o varias tecnologías, manteniendo en lo posible la definición de la interfaz.

1.3.1. Plataforma Moodle (EVA)

La plataforma Moodle dentro de entornos virtuales de aprendizaje (EVA) es un software de código libre para la creación de cursos y sitios web basados en la internet que está concebida y diseñada para que las personas que accedan a él, desarrollen procesos de incorporación de habilidades y saberes haciendo uso de sistemas telemáticos.

El Moodle por su facilidad de uso hace que sea una herramienta que pueda usarse en cualquier ámbito de la educación, con cualquier metodología docente y usuarios de cualquier nivel. Además se utiliza como soporte y apoyo para trabajos cooperativos. Promueve una pedagogía constructivista social (colaboración, actividades, reflexión, critica, etc.) al ser un entorno colaborativo con distintas funciones para la interacción y construcción del conocimiento de forma grupal, tiene una interfaz de navegación de tecnología sencilla, ligera y compatible, que requiere de una plataforma de PHP y en la mayoría de áreas de introducción de textos puede ser editado usando el editor HTML.

Las características que presenta dicha herramienta son:

- ✓ **Gran disponibilidad**: satisface las necesidades de profesores, estudiantes, administradores y creadores de contenidos.
- ✓ **Escalabilidad**: la aplicación se adapta a las necesidades que aparecen en el transcurso de la utilización de la misma. Tanto en organizaciones pequeñas como grandes se pueden utilizar la arquitectura Moodle.
- ✓ **Facilidad de uso**: las utilidades de Moodle son sencillas y su utilización es muy intuitiva. Existen manuales de ayuda que facilitan su utilización.
- ✓ **Interoperabilidad**: el código abierto propicia el intercambio de información gracias a la utilización de los “estándares abiertos de la industria para implementaciones web” (SOAP (Simple Object Access Protocol), XML…) Además se puede ejecutar en Linux, MacOS y Windows.

- ✓ **Estabilidad**: Moodle es un entorno eficaz y confiable.
- ✓ **Seguridad**: la restricción de acceso a las comunidades de aprendizaje de Moodle es una solución para evitar riesgos innecesarios (2).

Ventajas:

- ✓ Permite colocar recursos variados para formar una unidad de contenidos: etiquetas, archivos en formato variable (texto, audio, vídeo, hoja de cálculo, documento, presentación), web externas, edición de webs**.**
- ✓ Proporciona una información exhaustiva de la actividad de cada estudiante, minuto a minuto, día a día. Muestra el número de veces que entra, consulta, hace, aporta y realiza otras operaciones en las actividades propuestas. Permite el análisis de la información y la descarga de la misma a hoja de cálculo o documento de texto**.**
- ✓ La evaluación es continua y permanente (todo se comenta por todos y se evalúa)**.**
- ✓ Integra en una única pantalla información completa de manera útil y personalizada: quién está en línea, calendario, informe de la actividad reciente, mensajes, recursos o tareas añadidas al curso.
- ✓ Respecto al proceso de enseñanza y aprendizaje que permite un acercamiento a los temas desde muchos enfoques, con actividades múltiples y variadas que ponen en juego distintas capacidades (análisis, búsqueda y selección de información, elaboración de información, crítica…).
- ✓ Los estudiantes se familiarizan rápidamente con el entorno de la plataforma (3).

A pesar de las características y ventajas que presenta, se hace necesario tener una conexión permanente que posibilite el acceso a esta plataforma para satisfacer las necesidades de los usuarios en cualquier curso disponible, constituyendo así una limitación para quienes por diferentes razones no poseen acceso a la red.

1.3.2. Hot Potatoes.

Es un programa con el que se pueden crear actividades interactivas de carácter educativo fácilmente accesibles en línea a través de Internet. Es un software gratuito para uso individual o educativo sin ánimo de lucro, siempre y cuando el material producido sea accesible a través de Internet.

Hot Potatoes es un conjunto de seis herramientas que te permiten elaborar ejercicios interactivos basados en páginas web de seis tipos básicos. La información editable de cada tipo de ejercicio se guarda en un archivo específico. El alumno no necesita tener instalado en su equipo el programa para realizar cada ejercicio, sólo se requiere acceder utilizando un navegador como Internet Explorer u otro.

El sistema de administración del conocimiento para compartir documentos Hot Potatoes, utiliza la última tecnología wiki (sitio web colaborativo que puede ser editado por varios usuarios) y le permite insertar, crear y apagar contenido instantáneamente en una plataforma personalizable.

Características

- ✓ Es fácil de manejar, cualquier usuario sin saber nada de HTML o JavaScript, pero con algunos fundamentos básicos de informática puede manejar sus aspectos básicos en corto tiempo, creando páginas dinámicas que pueden colocarse en la Web. Para ello sólo tendrá que introducir los datos de los ejercicios.
- ✓ Permite exportar sus documentos al portapapeles y colocarlos en una aplicación como WORD.
- ✓ Permite enviar los resultados a una dirección de correo electrónico.
- ✓ Ofrece la posibilidad de añadir algunos elementos típicos de los tests dinámicos como puede ser incluir un feedback en cada pregunta.
- ✓ El programa acepta respuestas múltiples y permite incluir un reloj que limita el tiempo en el que la prueba se debe realizar.
- ✓ Permite barajar el orden de las preguntas y las respuestas cada vez que se carga. Evitando un aprendizaje mecánico de las mismas.
- ✓ La puntuación que ofrece tiene en cuenta, no sólo si la respuesta a una pregunta es correcta, sino también el número de intentos necesarios para responderla
- ✓ La puntuación final del cuestionario se calcula sumando los resultados de cada pregunta, los cuales se dividirán por el número de actividades. Los resultados se expresan en tanto por ciento.
- ✓ Hot Potatoes no se limita a generar cuestionarios de preguntas tipo test, también las presenta en forma de crucigramas mediante el módulo JCross.
- ✓ Las pruebas generadas por Hot Potatoes están más indicadas para la autoevaluación que para la evaluación externa. No deben utilizarse como instrumentos únicos de evaluación (4).

No se recomienda su utilización debido a que las actividades que genera no sustituyen un curso por su simplicidad conceptual y además el profesor no conoce los resultados del alumno en las pruebas generadas o en los cuestionarios realizados.

1.3.3. WebCT

WebCT es una herramienta virtual de gestión de curso similar a Moodle. Posee utilidades parecidas como: foros, chats, tablón de anuncios, contenido de cursos... Se creó en 1995 en la Universidad de Columbia

Británica en Canadá como un recurso para la creación de plataformas educativas basadas en páginas web (5).

Este entorno ha sido el preferido por las instituciones hasta ahora, pero debido a su elevado coste de mantenimiento ha provocado que numerosas universidades que hacen uso de ella migren sus cursos al entorno Moodle. Además obliga a los profesores y administradores de los cursos a usar modelos de navegación y formatos de cursos predeterminados. Estas limitaciones pueden tener un impacto negativo en la flexibilidad y usabilidad de la plataforma.

1.3.4. LRN

Esta plataforma que se pronuncia en inglés Dot Learn, es software libre educativo que da soporte a las comunidades de aprendizaje y de investigación. Está promovida por la Sloan School of Management del MIT y la Universidad de Heidelberg (5).

Se encuentra respaldada por numerosas instituciones educativas a nivel mundial, empresas y desarrolladores de código abierto, aunque su utilización no está muy extendida debido a que se encuentra en la fase inicial de su desarrollo.

1.4. Definición de Framework

En general, con el término framework, se hace alusión a una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se puede añadir las últimas piezas para construir una aplicación concreta.

Resulta complejo definir un concepto de framework, aunque cualquiera con experiencia en la programación puede ser capaz de elaborar su propio concepto. Algunas definiciones de framework son:

- ✓ "…diseño abstracto orientado a objetos para un determinado tipo de aplicación, que se compone de una clase abstracta para cada componente principal del diseño; contendrá normalmente una librería de subclases que pueden ser utilizadas como componentes del diseño…" (6).

- ✓ "…una mini-arquitectura reutilizable que provee la estructura genérica y el comportamiento para una familia de abstracciones de software, junto con un contexto formado por metáforas que especifican las colaboraciones y el uso en un dominio dado." (6).

En otras palabras, es una estructura de soporte bien definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado, entre otros, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

1.4.1. Ventajas de los framework

- ✓ **Mayor productividad del desarrollador:** al estar los frameworks enmarcados en un dominio, permiten que los desarrolladores sean productivos inmediatamente.
- ✓ **Menos codificación:** muchos de los códigos personalizados, ahora son reemplazados en su totalidad por el framework, o es una configuración del mismo; por lo que existe menos código.
- ✓ **Resultados inmediatos:** al dinamizar el proceso de asociación de los requerimientos a funcionalidades, se reduce considerablemente el tiempo de desarrollo. Esto permite un mayor número de iteraciones en un corto plazo, lo que conlleva a un producto de más calidad y mayor inmediatez en la entrega.
- ✓ **Mayor flexibilidad:** el bajo acoplamiento proporcionado por los frameworks aporta una gran flexibilidad, que responde a los cambios de los requisitos tanto del negocio como de tecnología.
- ✓ **Arquitectos más eficaces:** el esqueleto o estructura proporcionada por los frameworks permite que los arquitectos dediquen más tiempo a la toma de decisiones imprescindibles en vez de gastar una enorme cantidad de tiempo en hacer cumplir las mejores prácticas.

1.5. Descripción general del objeto de estudio

En la actualidad los frameworks son de vital importancia para construir grandes sistemas de software orientado a objeto, su desarrollo está ganando rápidamente una amplia aceptación debido a su capacidad para promover la reutilización del diseño y del código fuente, además simplifican el desarrollo de una aplicación mediante la automatización de algunos patrones utilizados para resolver las tareas comunes, así como proporcionan estructura a todos los documentos y códigos fuentes del sistema, forzando al equipo de desarrollo a crear documentos y códigos más legibles y más fáciles de mantener. Facilitan la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

La utilización de un framework en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo con seguridad facilitará tanto el desarrollo como el mantenimiento de la aplicación, además dinamiza el proceso de diseño y como los frameworks están alineados con los requisitos del negocio, el diseño se simplifica. Mejoran el desarrollo y las pruebas proporcionando los

siguientes beneficios: mayor productividad del desarrollador, menos codificación, mayor consistencia, resultados inmediatos, mayor flexibilidad. Si se usan de forma adecuada el software puede responder fácilmente a los cambios en los requerimientos tanto del negocio como de tecnología.

Por otra parte, las exigencias para la solución de problemas cada vez más complejos, resultan muy rigurosas e introducen premisas ineludibles al desarrollo de software:

- ✓ **Complejidad**: los sistemas cada vez son más y más complejos, lo que presupone que "a más líneas de código, más errores".
- ✓ **Extensibilidad**: los sistemas deben permitir incorporar funcionalidades sin necesidad de recompilar la totalidad del código.
- ✓ **Conectividad**: los sistemas deben estar preparados para interactuar con otros sistemas.
- ✓ **Reusabilidad**: los sistemas deben brindar la posibilidad de reutilizar software de otros proyectos o partes del mismo proyecto.

1.6. Metodologías y herramientas CASE a utilizar

Las metodologías o procesos de desarrollo de software ofrecen un método para la creación de aplicaciones flexibles y robustas de un modo organizado y disciplinado, facilitando su comprensión. Además abarcan procedimientos, técnicas, documentación y herramientas que tienen como base fundamental el fortalecimiento de los procesos de desarrollo de un software. Actualmente no existe una metodología universal que le haga frente con éxito a cualquier proyecto de desarrollo de software, ni que se adapte a todo tipo de sistema, debido a que las características y recursos de cada equipo de desarrollo no siempre son las mismas.

1.6.1. Metodologías tradicionales

Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. (7)

1.6.1.1. Proceso Unificado de Desarrollo (RUP)

RUP es un proceso formal: Provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, utilizando UML como lenguaje de modelado. Define cuatro fases esenciales (Inicio, Elaboración, Construcción y Transición) y nueve flujos de trabajos; seis de Ingeniería (Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue) y tres de apoyo.

Características de RUP:

- ✓ Establece un desarrollo iterativo.
- ✓ Permite la administración de requisitos.
- ✓ Hace uso de una arquitectura basada en componentes.
- ✓ Permite llevar un control de cambios.
- ✓ Permite realizar un modelado visual del software.
- ✓ Verifica la calidad del software.

Ventajas:

- ✓ Evaluación en cada fase que permite cambios de objetivos.
- ✓ Funciona bien en proyectos de innovación.
- ✓ Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- ✓ Seguimiento detallado en cada una de las fases.

Desventajas:

- ✓ El cliente debe ser capaz de describir y entender a un gran nivel de detalle para que los desarrolladores puedan acordar un alcance del proyecto con él.
- ✓ No es flexible para el desarrollo de proyectos pequeños que pueden estar sujetos a constantes cambios en sus requerimientos; debido a que un pequeño cambio en uno de ellos, provocaría que se tenga que generar nuevamente toda la documentación y prolongaría el tiempo de desarrollo del proyecto.

1.6.2. Metodologías ágiles

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance 3, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "ágil" (8).

1.6.2.1 SCRUM

SCRUM, más que una metodología de desarrollo software, es una forma de auto-gestión de los equipos de programadores. Esta metodología ayuda a los trabajadores a trabajar todos juntos, en la misma dirección, con un objetivo claro.

Principio de SCRUM

- ✓ Los equipos son auto-gestionados.
- ✓ Una vez dimensionadas las tareas no es posible agregarles trabajo extra.
- ✓ Se realizan reuniones diarias en las que los miembros del equipo se plantean 3 cuestiones:
  - ➢ ¿Qué has hecho desde la última revisión?
  - ➢ ¿Qué obstáculos te impiden cumplir la meta?
  - ➢ ¿Qué vas a hacer antes de la próxima reunión?
- ✓ Las iteraciones de desarrollo tienen una frecuencia inferior a un mes, al final de las cuales se presenta el resultado a los externos del equipo de desarrollo, y se realiza una planificación de la siguiente iteración, guiada por el cliente (9).

Ventajas:

- ✓ Entrega de un producto funcional al finalizar cada Sprint.
- ✓ Posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.

- ✓ Visualización del proyecto día a día.
- ✓ Alcance acotado y viable.
- ✓ Equipos integrados y comprometidos con el proyecto, toda vez que ellos definieron el alcance y se auto-administran (10).

1.6.2.2. Crystal

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (11).

Los roles definidos por Crystal son:

- ✓ Patrocinador Ejecutivo (Executive Sponsor)
- ✓ Jefe de Proyecto (Project Manager)
- ✓ Experto en el Dominio (Domain Expert)
- ✓ Experto de uso (Usage Expert)
- ✓ Programador Diseñador (Designer-Programmer)
- ✓ Diseñador UI (Designer)
- ✓ Realizador de Pruebas (Tester)
- ✓ Programador Técnico (Technical)

1.6.2.3. eXtreme Programming (XP)

XP es una metodología creada por Kent Beck, se caracteriza por retomar las prácticas de uso tradicional y llevarlas al extremo de ahí proviene su nombre. En busca de mejoras en el desarrollo de software y con bases fundamentadas de ingeniería de software, esta metodología detalla varios valores de los métodos ágiles.

Se basa en la **realimentación** continua entre el cliente y el equipo de desarrollo, la **comunicación** fluida entre todos los participantes, la **simplicidad** en las soluciones implementadas y el **coraje** para enfrentar los cambios.

Básicamente, la metodología XP se basa en cinco valores:

- ✓ **Comunicación:** La comunicación permanente es fundamental en XP. Dado que la documentación es escasa, el diálogo frontal, cara a cara, entre desarrolladores, gerentes y el cliente es el medio básico de comunicación. Una buena comunicación tiene que estar presente durante todo el proyecto.
- ✓ **Simplicidad:** La sencillez es esencial para que todos puedan entender el código, y se trata de mejorar mediante recodificaciones continuas.
- ✓ **Feedback:** Básicamente el continuo contacto con el usuario, al irle entregando las sucesivas versiones, en funcionamiento del producto, permite que este nos dé su valoración y nos comunique, cada vez mejor, lo que realmente quiere en el producto.
- ✓ **Coraje:** Básicamente es trabajar muy duro durante las horas dedicadas a ello.
- ✓ **Respeto:** Si los miembros de un equipo no se preocupan por sí mismos y por su trabajo, la metodología no puede funcionar. Es necesario ser respetuoso con sus colegas, sus contribuciones, su organización y con las personas cuya vida se toca por el sistema que está escribiendo (12).

Sustentados en estos 5 valores los proyectos XP deben tener muy presentes 24 prácticas, que se integran unas con otras y que ofrecen una base consistente para un recomendable desempeño, alta productividad e incalculables beneficios. Entre ellas destacamos:

- ✓ Ciclo semanal
- ✓ Programación en pares
- ✓ Incremento del Diseño
- ✓ Primer test Programación
- ✓ Integración continua
- ✓ Participación real de clientes

El ciclo de vida ideal consta de 6 fases:

1. **Exploración:** En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se

familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

2. **Planificación de Entregas:** En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.
3. **Iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas.
4. **Producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente.
5. **Mantenimiento:** Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente.
6. **Muerte:** Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura (12).

1.6.3. Justificación de la metodología seleccionada

Una vez finalizado el estudio realizado sobre las posibles metodologías de desarrollo de software que podrían haber guiado el proceso de desarrollo del framework, se seleccionó por parte del equipo de trabajo siguiendo las políticas establecidas por el proyecto “Plataforma Educativa Dolphin”, la metodología ágil XP, descartando las anteriormente analizadas principalmente por las desventajas encontradas durante el análisis de cada una de ellas, debido a que se debían tener en cuenta algunas necesidades por parte de los desarrolladores, para la realización de la solución propuesta; en primer lugar una de las características fundamental de XP, es que el cliente forme parte del equipo de desarrollo, requisito que se cumple debido a la cercanía constante del cliente; por otra parte, el equipo de trabajo cuenta con dos integrantes, la dimensión del proyecto es pequeña y está basado en el uso de nuevas tecnologías por lo que se suman los riesgos técnicos asociados a ellas, la planificación se realiza a corto plazo porque está sujeta a las necesidades de la producción donde se exigen entregas en breves intervalos lo cual es perfectamente posible gracias a su naturaleza iterativa e incremental y por último se consigue tener un proceso de desarrollo motivado, ya que XP no permite excesos de trabajos y se consigue una mayor

integración entre los miembros por la comunicación establecida, además está probada en disimiles proyectos y tiene suficiente soporte y documentación accesible en la red.

1.6.4. Herramientas CASE

CASE es una sigla, que corresponde a las iniciales de: Computer Aided Software Engineering; y en su traducción al Español significa Ingeniería de Software Asistida por Computadoras. Las herramientas CASE representan una forma que permite Modelar los Procesos de Negocios de las empresas, tienen como objetivo fundamental solucionar y afrontar los problemas de una mala calidad de software y una documentación inadecuada. Además de conseguir la generación automática de programas desde una especificación a nivel de diseño.

1.6.4.1. Rational Rose

Rose es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Una de las grandes ventajas de Rose es que utiliza la notación estándar en la arquitectura de Software (UML), la cual permite a los arquitectos de software y desarrolladores visualizar el sistema completo utilizando un lenguaje común. Otra ventaja de la herramienta es que los diseñadores pueden modelar sus componentes e interfaces en forma individual y luego unirlos con otros componentes del proyecto.

Características

- ✓ Mantiene la consistencia de los modelos del sistema software.
- ✓ Chequeo de la sintaxis UML.
- ✓ Generación automática de la documentación.
- ✓ Generación de código a partir de los modelos: se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.
- ✓ Ingeniería inversa: crear modelo a partir código (13).

Aunque esta herramienta es una buena elección para el ambiente de modelado, no genera código para ActionScript 3.0, por lo que no cumple con las necesidades requeridas para la realización del framework.

1.6.4.2 Visual Paradigm 6.04

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El

software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML.

Esta herramienta tiene las siguientes características:

- ✓ Soporte de UML versión 2.1.
- ✓ Modelado colaborativo con CVS y Subversion (nueva característica).
- ✓ Interoperabilidad con modelos UML 2 (metamodelos UML 2.x para plataforma Eclipse) a través de XMI (nueva característica).
- ✓ Ingeniería inversa - Código a modelo, código a diagrama.
- ✓ Ingeniería inversa Java, C++, Esquemas XML, XML, NET exe/dll, CORBA IDL.
- ✓ Generación de código - Modelo a código, diagrama a código.
- ✓ Editor de Detalles de Casos de Uso - Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- ✓ Diagramas EJB - Visualización de sistemas EJB.
- ✓ Generación de código y despliegue de EJB´s - Generación de beans para el desarrollo y despliegue de aplicaciones.
- ✓ Diagramas de flujo de datos.
- ✓ Soporte ORM - Generación de objetos Java desde la base de datos.
- ✓ Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- ✓ Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- ✓ Generador de informes para generación de documentación, Distribución automática de diagramas, Reorganización de las figuras y conectores de los diagramas UML.
- ✓ Importación y exportación de ficheros XML.
- ✓ Integración con Visio - Dibujo de diagramas UML con plantillas (stencils) de MS Visio (14).

Visual Paradigm 6.04 para UML, además de ser una herramienta que posee licencia gratuita y comercial, es la herramienta case que nos permite llevar una buena organización y planificación de los diagramas y

modelados del framework a desarrollar, así como genera código para ActionScript 3.0, lo cual es de vital importancia para el desarrollo de la propuesta de solución.

1.7. UML. El Lenguaje de Modelado Unificado

UML es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

Características del UML

- ✓ UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. Está basado en el común acuerdo de gran parte de la comunidad informática.
- ✓ UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. UML incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
- ✓ Debe ser tan simple como sea posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. UML necesita ser lo suficientemente expresivo para manejar todos los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.
- ✓ Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
- ✓ Pretende imponer un estándar mundial.
- ✓ Es orientado a objetos, permitiéndole al programador que organice su programa de acuerdo con abstracciones de más alto nivel, siendo estas más cercanas a la forma de pensar de las personas (15).

Los principales beneficios de UML son:

- ✓ Mejores tiempos totales de desarrollo (de 50 % o más).
- ✓ Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- ✓ Establecer conceptos y artefactos ejecutables.
- ✓ Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- ✓ Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- ✓ Mejor soporte a la planeación y al control de proyectos.
- ✓ Alta reutilización y minimización de costos (16).

1.8. Lenguaje de programación

ActionScript fue lanzado con la versión 4 de Flash, y desde entonces hasta ahora, ha ido extendiéndose poco a poco, hasta llegar a niveles de dinamismo y versatilidad muy altos en la versión 10 (Adobe Flash CS4) de Flash. Además es un lenguaje de programación orientado a objetos (POO), y es utilizado principalmente en aplicaciones Web realizadas en el ambiente Adobe Flash.

La versión más extendida actualmente es ActionScript 3.0 (AS 3.0), que significó una mejora en el manejo de programación orientada a objetos y es utilizada en las últimas versiones de Adobe Flash y Flex y en anteriores versiones de Flex. AS 3.0 incluye nuevas características como el uso de expresiones regulares y nuevas formas de empaquetar las clases.

¿Por qué ActionScript 3.0 como lenguaje de programación?

AS 3.0 es un potente lenguaje de POO que ofrece un modelo de programación robusto. Este aumenta las posibilidades de creación de scripts (guión o conjunto de instrucciones) de las versiones anteriores, facilita la creación de aplicaciones muy complejas con conjuntos de datos y bases de código reutilizables y orientadas a objetos. No requiere necesariamente de Adobe Flash Player 9 para la ejecución de su contenido.

Una de las desventajas que corrige AS 3.0 con respecto a las versiones anteriores, es el tratamiento de eventos, ya que contiene en su framework un paquete de clases exclusivamente dedicadas a su gestión. Además respecto a versiones anteriores AS 3.0 notifica más situaciones de error en tiempo de ejecución permitiendo la depuración así como la identificación rápida de los errores.

Los objetivos que se persiguen con este lenguaje se reúnen en los siguientes términos:

- ✓ **Seguridad:** El código será más fácil de mantener, eliminando ambigüedades y permitiendo una escritura más clara y concisa
- ✓ **Simplicidad:** El lenguaje será lo suficientemente intuitivo como para que el programador pueda escribir código sin tener que consultar el manual de referencia constantemente.
- ✓ **Performance:** El lenguaje habilitará a los programadores para escribir complejos programas que funcionen de forma más eficiente y responsable.
- ✓ **Compatibilidad:** Como dialecto de la especificación ECMAScript, el lenguaje estará provisto de una mayor compatibilidad con los estándares de la industria. Se pretende lograr un lenguaje más coherente y unificado (17).

Estas son algunas de las nuevas características del lenguaje de Flash:

- ✓ La implementación de una renovada y altamente optimizada "máquina virtual" (AVM2: Actionscript Virtual Machine 2) supone un drástico incremento en la performance del lenguaje, superando ampliamente a la máquina virtual original (AVM1), responsable de las versiones 1 y 2. Como resultado de este incremento, Actionscript 3.0 ejecuta hasta 10 veces más rápido el código, en comparación con sus versiones predecesoras.
- ✓ Soporte para expresiones regulares (esta es quizá una de las funcionalidades más reclamadas en la comunidad de programadores). Las expresiones regulares son funciones que permiten el ajuste de una cadena de texto a un determinado patrón, permitiendo por ejemplo validar los campos de un formulario, o implementar sistemas de búsqueda y reemplazo de cadenas.
- ✓ XML es ahora un tipo de dato nativo de Actionscript gracias a la tecnología E4X, extensión que ofrece soporte nativo de XML para lenguajes que cumplen con la especificación ECMAScript.
- ✓ Un nuevo modelo para el manejo de eventos, claramente orientado a objetos.
- ✓ El uso de sockets binarios, que permitirá a Flash comunicarse con nuevos protocolos, abriendo todo un campo de posibilidades para el desarrollo de aplicaciones online.
- ✓ Manejo de errores en tiempo de ejecución, lo cual facilitará la depuración y el control de los scripts (17).

Por tanto si decide desarrollar el framework en el lenguaje AS 3.0 debido a las ventajas que brinda el mismo a la programación orienta a objeto.

1.9. Herramientas a utilizar

1.9.1. Balsamiq Mockup

El programa está pensado para la creación rápida y eficaz de cualquier tipo de Maqueta, tanto de proyectos web como de software. Entre sus características destaca el estilo de los componentes realizados imitando a los que podríamos dibujar nosotros en un papel pero eliminando ese paso y haciéndolo directamente en la PC.

Está desarrollada sobre Flex y cuenta con una versión en línea. La app desktop está portada sobre AIR 1.5 e incorpora las funcionalidades más recientes de ActionScript 3.0. Exporta ficheros PDF (generados con AlivePDF), imágenes PNG y un fichero XML con el “Mockup” para posteriormente importarlo. La gestión propia de ficheros lo hace a partir de archivos XML de extensión “bmml” (18).

1.9.2. TortoiseSVN 1.4.5

Un cliente de Subversion, implementado como una extensión del shell de Windows. TortoiseSVN es muy fácil de utilizar en el control de revisión y control de versiones. Es software libre liberado bajo la licencia GNU GPL. Puede ser usado sin un entorno de desarrollo y está disponible en 28 idiomas diferentes. Decora los íconos con pequeñas imágenes mostrando qué archivos o directorios necesitan ser enviados al repositorio.

1.9.3. Adobe Flex Builder 3

El software Adobe Flex Builder 3 Professional es un entorno de desarrollo integrado basado en Eclipse para la creación de aplicaciones dinámicas de Internet (RIA) fundamentales para el ámbito empresarial. Presenta una serie de funciones como la codificación inteligente, la depuración, el diseño visual de la interfaz de usuario, los componentes avanzados de gráficos, los perfiles y la compatibilidad con las pruebas, estas funcionalidades automatizadas hacen de Flex Builder 3 Professional la mejor opción para el desarrollo empresarial.

Flex 3 es un programa que permite utilizar aplicaciones de Internet bastante complejas en el escritorio, con funcionalidades como arrastrar y soltar, empleo de múltiples ventanas, acceso al sistema de archivos locales, almacenamiento en la base de datos SQLite local, interpretación HTML completa, predicción de códigos AIR e informes de errores completos, creación de perfiles y depuración de AIR y agrupación y firma de la aplicación AIR.

El lenguaje de Flex se basa en estándares modernos, con un modelo de programación que admite los patrones de diseño habituales. MXML es un lenguaje declarativo basado en XML, y empleado para describir el aspecto y comportamiento de la interfaz de usuario; por su parte ActionScript 3, es un potente lenguaje de programación que permite manejar objetos. Flex incorpora una biblioteca de componentes muy completa con más de 100 componentes de interfaz de usuario extensible y de eficacia demostrada para crear RIA, así como un depurador interactivo de aplicaciones de Flex (19).

Las razones principales para utilizar Adobe Flex Builder 3 son:

- ✓ Eficaces herramientas de codificación.
- ✓ Sofisticado diseño visual.
- ✓ Aplicación de aspectos y estilos.
- ✓ Integración con Adobe Creative Suite 3.

- ✓ Refactorización del código.
- ✓ Compatibilidad original con Adobe AIR.
- ✓ Servicios avanzados de datos.
- ✓ Compatibilidad con el kit de desarrolladores de software de Flex 2 y 3 (20).

1.10. Conclusiones

En el presente capítulo se realizó una investigación sobre el estado del arte de los distintos frameworks existentes para la generación de interfaces gráficas para plataformas educativas determinándose que ninguna de las soluciones existentes se ajusta a las necesidades, ya que la mayoría están sujetas a restricciones como: coste de mantenimiento, entorno visual, administración del sistema o simplemente se encuentra en la fase inicial de su desarrollo. Después de realizar el análisis de las metodologías existentes se decidió que por sus facilidades, documentación, y flexibilidad se utilizaría XP y además para la realización de la implementación del framework se determinó utilizar Adobe Flex Builder 3 Professional por ser una herramienta que brinda una serie de funcionalidades para el trabajo con ActionScript 3.0. Como herramienta CASE se seleccionó Visual Paradigm 6.04 apoyado en UML para el modelado del sistema.

Características del Sistema. Exploración y Planificación

2.1. Introducción

El presente capítulo tiene como objetivo hacer una valoración de las principales características del framework a desarrollar. Se hace mención a las fases de exploración y planificación, propias de la metodología de desarrollo utilizada para la implementación del framework que se propone y se exponen los artefactos generados durante el transcurso de las mismas.

2.2. Descripción de los procesos vinculados al campo de acción

Actualmente, en la Universidad de las Ciencias Informáticas en el proceso de desarrollo de frameworks, no existe el diseño de una arquitectura que sirva de plantilla, además que permita la reutilización de código y componentes, lo que contribuye a que el tiempo de desarrollo de las plataformas educativas se extienda, provocando un aumento en los costos de producción y un mayor uso del recurso humano.

2.3. Propuesta del sistema

El presente trabajo propone el proceso de desarrollo de un framework basado en la utilización de la herramienta Flex Builder 3 como editor de código, así como un estándar de desarrollo de software bajo plantillas y documentación, que representan un factor fundamental en los métodos de trabajos colectivos e individuales de cada estructura de la misma.

El Framework es la base de las aplicaciones Dolphin. Compuesto por un gran número de clases relacionadas y reutilizables, diseñadas para proporcionar funcionalidades útiles de propósito general.

Basado en su importancia y uso, las clases del framework han sido clasificadas en las siguientes categorías:

- ✓ **Interactividad:** útiles para la creación de actividades interactivas, ya sean selección simple o múltiple, entre otros. Todo esto vinculado a documentos XML.
- ✓ **Estilos de Información:** encargados de la representación visual del contenido y de lograr un formato uniforme respecto al entorno y a los objetos/textos del producto. Personalizable desde hojas de estilo (CSS) para los textos.

- ✓ **Gestión de Recursos:** representa la base del entorno y propicia la información completa de las rutas de las imágenes, textos, videos, sonidos. Resulta muy útil a la hora de la creación de plataformas educativas y entornos similares, ya que facilita la modificación de las rutas de los recursos.
- ✓ **Componentes:** acompañados de clases, aunque no las tienen como principal objetivo. Son compontes prefabricados a partir del trabajo en clases como símbolos, botones, menús, ventanas, etc. (15).

La documentación de Dolphin ofrece la posibilidad de obtener información precisa de los métodos de las clases, especificaciones de éstas y ejemplos concretos de código en formato *.swf. Presenta la base de uso del framework y los detalles de trabajo de sus clases de forma colectiva e independiente.

Las plantillas son un medio que permite guiar, portar o construir un diseño o esquema predefinido de las medias que conforman la plataforma educativa Dolphin. Están formadas por cuatro partes fundamentales:

- ✓ Plantilla de actividad interactiva de selección simple.
- ✓ Plantilla de actividad interactiva de selección múltiple.
- ✓ Plantilla de reproductor de sonido y contenidos (textos e imágenes).
- ✓ Plantilla de reproductor de video y contenidos (textos e imágenes).

2.3.1. Personal relacionado con el sistema

Se define como persona relacionada con el sistema a aquella que está de una forma u otra vinculada al proceso de desarrollo del framework, así como las que interactúan con el mismo.

Tabla 2.3.1 Personas relacionadas con el sistema

Personas relacionadas con el sistema.	Justificación
Desarrollador	Es la persona encargada de realizar la implementación de nuevas funcionalidades al framework.
Rol de usuario	• Profesor: encargado de generar los contenidos que conforman los cursos de la plataforma educativa. • Estudiantes: usuario que interactúa con los contenidos visuales generados.

2.4. Fase de exploración

La metodología de desarrollo XP comienza con su fase de exploración, durante esta etapa se define el alcance general del proyecto, además los clientes plantean a grandes rasgos todas las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Las estimaciones realizadas en esta fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más en detalle en cada iteración. Esta fase toma de pocas semanas a pocos meses, dependiendo de la familiaridad que tengan los programadores con la tecnología.

2.4.1. Historias de usuarios

Las Historias de usuario (HU) son utilizadas en la metodología de desarrollo ágil XP para representar una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada funcionalidad del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, reemplazan un gran documento de requisitos y presiden la creación de las pruebas de aceptación.

La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Deben poder ser programadas en un tiempo entre una y tres semanas. Si la estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia de usuario.

Las Historias de Usuario tienen tres aspectos:

- ✓ **Tarjeta:** en ella se almacena suficiente 1 para identificar y detallar la historia.
- ✓ **Conversación:** cliente y programadores discuten la historia para ampliar los detalles (verbalmente cuando sea posible, pero documentada cuando se requiera confirmación).
- ✓ **Pruebas de Aceptación:** permite confirmar que la historia ha sido implementada correctamente.

Plantilla de Historias de Usuarios: (15)

Historia de Usuario	
No.: Número sucesivo a partir de 1.	**Nombre:** Identifica la historia de usuario en cuestión.
Usuario: Quien ejecuta la historia de usuario.	
Prioridad en el Negocio: Define la relevancia e impacto de la historia de usuario para el negocio de acuerdo con las necesidades del usuario.	**Nivel de Complejidad:** Define la dificultad técnica que supone desarrollar la historia de usuario desde el punto de vista del programador.
Puntos de Estimación: Permiten estimar duración de implementación.	**Iteración Asignada:** Precisa la iteración a la que pertenece la historia de usuario.
Descripción: Explica en qué consiste la historia de usuario, teniendo en cuenta las acciones realizadas por el usuario y la respuesta brindada por el sistema.	
Información adicional (Observaciones): Información extra que se estime agregar para hacer más comprensible la historia de usuario. Por ejemplo: conceptos, post-condiciones, relación con otros requisitos, etc.	

Como resultado del trabajo realizado durante las fases de exploración se identificaron las siguientes historias de usuario:

Tabla 2.4.1 HU Generación de XML

Historia de Usuario	
No.: 1	**Nombre:** Generación de XML.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Permite generar archivos XML para cada una de las medias (textos, imágenes, videos, sonidos, juegos, actividades interactivas), mediante una clase que se encarga de parsear el MXML.	
Información adicional (Observaciones): Al generar un XML permite: Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.	

Tabla 2.4.2 HU Carga de XML

Historia de Usuario	
No.: 2	**Nombre:** Carga de XML.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Permite cargar los archivos XML de cada una de las medias ya generadas, mediante de una clase que se encarga de parsear el XML.	
Información adicional (Observaciones): Cualquier aplicación que trabaje con XML necesita un módulo de clases, su función es leer documentos y proporcionar acceso a su contenido y estructura. Para poder llevar a cabo esta función, la aplicación debe proporcionar información al procesador XML de cómo se encuentra almacenada esta información a través de un DTD. El DTD o declaración del tipo de documento (Document Type Declaration) proporciona la gramática para una clase de documentos XML.	

Tabla 2.4.3 HU Reproductor de Audio

Historia de Usuario	
No.: 3	**Nombre:** Reproductor de Audio.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: En caso de que la aplicación tenga un reproductor de audio, este debe permitir realizar funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento.	
Información adicional (Observaciones): El reproductor debe contar además con un temporizador que indique la longitud del archivo y la posición actual del cabezal del reproductor. Se hace referencia a las funcionalidades: crear reproductor, reproducir audio, pausar audio, bajar volumen del audio, seleccionar una posición para reproducir el audio, crear un temporizador.	

Tabla 2.4.4 HU Reproductor de Video

Historia de Usuario	
No.: 4	**Nombre:** Reproductor de Video.
Usuario: Desarrollador	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: En caso de que la aplicación tenga un reproductor de video, este debe permitir realizar funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento.	
Información adicional (Observaciones): El reproductor debe contar además con un temporizador que indique la longitud del archivo y la posición actual del cabezal del reproductor. Se hace referencia a las funcionalidades: crear reproductor, reproducir película, pausar película, bajar volumen de la película, seleccionar una posición para reproducir la película, crear un temporizador.	

Tabla 2.4.5 HU Actividad Interactiva de Selección

Historia de Usuario	
No.: 5	**Nombre:** Actividad Interactiva de Selección
Usuario: Rol de usuario	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 3
Descripción: Se le debe brindar la posibilidad, a cualquier persona interesada en medir sus habilidades y conocimientos adquiridos mediante el estudio de los contenidos del producto, que lo puedan hacer a través de ejercicios interactivos que contribuyan a un mejor aprendizaje de los contenidos. En cada una de las distintas tipologías de ejercicio propuestas, el sistema debe informar al usuario si dicha actividad ha sido realizada satisfactoriamente, una vez termine el ejercicio.	
Información adicional (Observaciones): En algunas tipologías, como por ejemplo: selección simple y selección compuesta, el sistema debe brindarle la posibilidad al usuario de volver a intentar el ejercicio, en caso de que se haya equivocado anteriormente. Se hace referencia a las funcionalidades: crear actividades interactivas de selección simple, crear actividades interactivas selección compuesta, validar respuestas de actividades interactivas.	

2.5. Fase de planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

2.5.1. Estimación de esfuerzos por historia de usuario

Para el buen desarrollo del sistema propuesto, se realizó una estimación para cada una de las historias de usuario identificadas, llegando a los resultados que se muestran a continuación:

Tabla 2.5.1 Estimación de esfuerzo por historia de usuario

Historias de usuario	Puntos de estimación
Generación de XML.	1 semana
Carga de XML.	1 semana
Reproductor de Audio.	1 semana
Reproductor de Video.	1 semana
Actividad Interactiva de Selección.	1 semana

2.5.2. Plan de iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteraciones son: historias de usuario. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Una vez definidas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se decide realizar el sistema en 3 iteraciones, las cuales se describen detalladamente a continuación:

Iteración 1

Esta iteración tiene como objetivo darle cumplimiento a las HU que se consideraron de mayor importancia para el desarrollo del framework. Al concluir dicha iteración se contará con todas las funcionalidades descritas en las HU 1 y 2, las cuales hacen alusión a todo lo referente con la generación y carga de XML.

Iteración 2

Esta iteración tiene como finalidad desarrollar las HU que responden a los números 3 y 4. Dichas HU son las que brindan las funcionalidades de reproducción de audio y video a través de una galería de audio y video. Además permiten a los usuarios controlar el estado de la reproducción de los contenidos, mediante algunas funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento. La versión que se obtendrá de esta iteración en unión con la entregada en la iteración anterior se le dará al cliente para verificar si cumple con las necesidades antes acordadas con él.

Iteración 3

Esta es la última iteración del framework, en la que se dará cumplimiento a las HU 5. Dicha HU cumple con las funcionalidades de crear actividades interactivas de selección simple, selección compuesta, además permite validar las respuestas de actividades interactivas. Esta HU es integrada con el resultado de las iteraciones anteriores y como fruto de esta integración se obtendrá la versión 1.0 del framework, que contiene cuatro plantillas, que muestran la utilidad del framework y la arquitectura desarrollada. A partir de este momento el framework será puesto a un proceso de prueba para evaluar el desempeño del mismo.

2.5.3. Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo de software XP, se crea el plan de duración de cada una de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas.

Tabla 2.5.3 Plan de duración de las iteraciones

Iteración	Historias de Usuario	Duración total iteraciones
Iteración 1	Generación de XML.	2 semanas
	Carga de XML.	
Iteración 2	Reproductor de Audio.	2 semanas
	Reproductor de Video.	
Iteración 3	Actividad Interactiva de Selección.	1 semana

2.6. Conclusiones

En este capítulo se inicia el desarrollo de la propuesta de solución que se desea implementar, se analizó la propuesta del sistema. Además se trata todo lo referente a las dos primeras fases de la metodología de desarrollo de software a utilizar, fase de exploración y planificación del sistema, donde se documentaron todos los artefactos generados en el transcurso de las mismas. Y quedó plasmado que el desarrollo del framework se realizará en 3 iteraciones y programado en pareja, como lo propone la metodología utilizada.

Construcción de la Solución Propuesta

3.1. Introducción

La Metodología XP plantea que la implementación de un software debe realizarse de forma iterativa, obteniendo al culminar cada iteración un producto funcional que debe ser probado y mostrado al cliente para incrementar la visión de los desarrolladores con la opinión de éste.

En el presente capítulo se presenta el diagrama de clases de la arquitectura del framework, así como el estándar de codificación utilizado para la implementación de las clases de Dolphin. Además se detallan las tres iteraciones llevadas a cabo durante la etapa de construcción del sistema, exponiéndose las tareas generadas por cada historia de usuario y las pruebas de aceptación efectuadas sobre el sistema.

3.2. Diseño de la solución propuesta

La metodología de desarrollo XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo del software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

Para el diseño de las aplicaciones, la metodología XP no requiere la presentación del sistema mediante diagramas de clases utilizando notación UML, en su lugar se usan otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante el uso de estos diagramas puede aplicarse siempre y cuando influyan en el mejoramiento de la comunicación entre el equipo de desarrollo, no sea un peso su mantenimiento, no sean extensos y se enfoquen en la información importante.

3.2.1. Tarjetas CRC

Para poder diseñar el sistema como un equipo, se debe cumplir con tres principios: Cargo o Clase, Responsabilidad y Colaboración (CRC). Las tarjetas CRC permiten desprenderse del método de trabajo basado en procedimientos y trabajar con una metodología basada en objetos. Además permiten que el equipo completo contribuya en la tarea del diseño.

Debido a la facilidad de uso y entendimiento, que propician dichas tarjetas, el equipo de desarrollo del proyecto “Plataforma Educativa Dolphin”, decidió utilizarlas para diseñar el framework que se desea desarrollar.

Tabla 3.2.1 Plantilla para las Tarjetas CRC

Tarjeta CRC	
Clase: Nombre de la clase que se está modelando.	
Súper Clase: Nombre de la clase padre en la herencia.	
Sub Clase(s): Nombre de la(s) clase(s) hija en la herencia.	
Responsabilidades: Es una descripción de alto nivel del propósito de la clase.	**Colaboraciones:** Indica con cuáles otras clases se requiere relación para cumplir la responsabilidad.

Tabla 3.2.2 Tarjeta CRC DCoreParsing

Tarjeta CRC	
Clase: DCoreParsing	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Parsear un XML. Generar un mxml con la misma estructura definida en el XML parseado.	**Colaboraciones:** Clase DRadioButton Clase DText Clase DImage Clase DCheckBox Clase DComboBox Clase DTextArea

Tabla 3.2.3 Tarjeta CRC DCoreInverseParsing

Tarjeta CRC	
Clase: DCoreInverseParsing	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Generar un XML con la misma estructura definida en el mxml parseado.	**Colaboraciones:** Clase DRadioButton Clase DText Clase DImage Clase DCheckBox Clase DComboBox Clase DTextArea

Tabla 3.2.4 Tarjeta CRC DComponent

Tarjeta CRC	
Clase: DComponent	
Súper Clase: mx.core.UIComponent	
Sub Clase(s): -	
Responsabilidades: Comprobar si un atributo está definido en una etiqueta de un XML. Comprobar si el mismo posee algún valor.	**Colaboraciones: -**

Tabla 3.2.5 Tarjeta CRC DContainer

Tarjeta CRC	
Clase: DContainer	
Súper Clase: mx.containers.Box	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente Box que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.6 Tarjeta CRC DRadioButton

Tarjeta CRC	
Clase: DRadioButton	
Súper Clase: mx.controls.RadioButton	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente RadioButton que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.7 Tarjeta CRC DText

Tarjeta CRC	
Clase: DText	
Súper Clase: mx.controls.Text	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente Text que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.8 Tarjeta CRC DImage

Tarjeta CRC	
Clase: DImage	
Súper Clase: mx.controls.Image	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente Image que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.9 Tarjeta CRC DCheckBox

Tarjeta CRC	
Clase: DCheckBox	
Súper Clase: mx.controls.CheckBox	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente CheckBox que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.10 Tarjeta CRC DComboBox

Tarjeta CRC	
Clase: DComboBox	
Súper Clase: mx.controls.ComboBox	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente ComboBox que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.11 Tarjeta CRC DTextArea

Tarjeta CRC	
Clase: DTextArea	
Súper Clase: mx.controls.TextArea	
Sub Clase(s): -	
Responsabilidades: Asignar valor a los atributos que posee el componente TextArea que se va a generar automáticamente. Estos valores son recogidos de un XML.	**Colaboraciones:** Clase DComponent

Tabla 3.2.12 Tarjeta CRC DMediaPlayer

Tarjeta CRC	
Clase: DMediaPlayer	
Súper Clase: -	
Sub Clase(s): DVideoPlayer, DAudioPlayer	
Responsabilidades: Crear reproductor con todas las funcionalidades.	**Colaboraciones: -**

Tabla 3.2.13 Tarjeta CRC DVideoPlayer

Tarjeta CRC	
Clase: DVideoPlayer	
Súper Clase: DMediaPlayer	
Sub Clase(s): -	
Responsabilidades: Permitir reproducir el video. Permitir opciones de video.	**Colaboraciones: -**

Tabla 3.2.14 Tarjeta CRC DAudioPlayer

Tarjeta CRC	
Clase: DAudioPlayer	
Súper Clase: DMediaPlayer	
Sub Clase(s): -	
Responsabilidades: Permitir reproducir el audio. Permitir opciones de audio.	**Colaboraciones: -**

Tabla 3.2.15 Tarjeta CRC DEjercicio

Tarjeta CRC	
Clase: DEjercicio	
Súper Clase: mx.containers.Box	
Sub Clase(s): DSeleccion	
Responsabilidades: Crear un ejercicio. Mostrar la información de un ejercicio. Actualizar la información de un ejercicio. Eliminar un ejercicio. Generar los ficheros XML.	**Colaboraciones: -**

Tabla 3.2.16 Tarjeta CRC DSeleccion

Tarjeta CRC	
Clase: DSeleccion	
Súper Clase: DEjercicio	
Sub Clase(s): -	
Responsabilidades: Visualizar enunciado del ejercicio. Visualizar tipo de Interfaz del ejercicio.	**Colaboraciones: -**

Tabla 3.2.17 Tarjeta CRC DInciso

Tarjeta CRC	
Clase: DInciso	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Crear un inciso. Mostrar la información de un inciso. Actualizar la información de un inciso. Eliminar un inciso.	**Colaboraciones: -**

Tabla 3.2.18 Tarjeta CRC DMotorGrafico

Tarjeta CRC	
Clase: DMotorGrafico	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Graficar un ejercicio determinado devolviendo el objeto de UIComponent que se mostrará.	**Colaboraciones: -**

3.2.2. Arquitectura del sistema

Una arquitectura de software, también denominada arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La arquitectura de software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. Además define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos (21).

Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información (21).

La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación y tiene la responsabilidad de:

- ✓ Definir los módulos principales.
- ✓ Definir las responsabilidades que tendrá cada uno de estos módulos.
- ✓ Definir la interacción que existirá entre dichos módulos:
  - Control y flujo de datos.

- Secuenciación de la información.
- Protocolos de interacción y comunicación.
- Ubicación en el hardware (22).

Para el desarrollo del sistema propuesto se tomó en cuenta el estilo de programación por capas, cuyo objetivo primordial es la separación de la capa lógica del negocio de la lógica del diseño, un ejemplo básico de esto es separar la capa de datos de la capa de presentación al usuario.

Ventajas de esta Arquitectura:

- ✓ El desarrollo se puede llevar a cabo en varios niveles.
- ✓ Desarrollos paralelos (en cada capa).
- ✓ Aplicaciones más robustas debido al encapsulamiento.
- ✓ En caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
- ✓ Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- ✓ Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad) (15).

3.2.2.1. Diagrama de componentes

Un diagrama de componentes muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables, ilustran las piezas del software, controladores embebidos, etc. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema, es decir, para describir la vista de implementación estática de un sistema. Los diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones pero un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clases, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución (23).

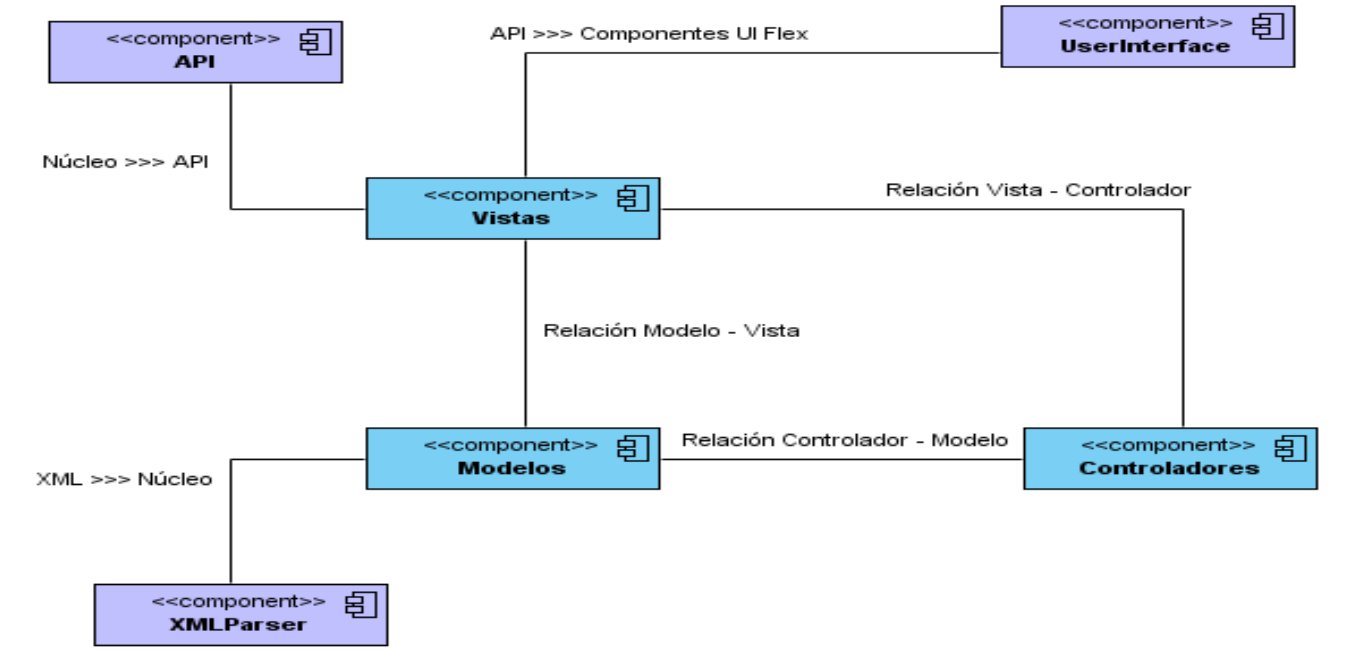

Figura 3.1 Diagrama de Componentes del Framework de la API de Dolphin.

Tabla 3.1 Componentes del framework de la API de Dolphin.

Nombre	Documentación
API	Componente principal del núcleo del sistema "Dolphin". Es el encargado de generar los componentes de interfaz gráfica de la Vista.
UserInterface	Interfaz gráfica del sistema. Se subordina a la Vista y porta el contenido generado desde la API.
Vistas	Conjunto de Vistas del patrón Modelo-Vista-Controlador.
Modelos	Conjunto de Modelos del patrón Modelo-Vista-Controlador.
Controladores	Conjunto de Controladores del patrón Modelo-Vista-Controlador.
XMLParser	Componente auxiliar de parseo de XML para el Modelo. Es utilizado directamente para generar el sistema de clases del núcleo a partir de las peticiones al servidor.

3.2.3. Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

Un patrón de diseño es:

- ✓ Una solución estándar para un problema común de programación.
- ✓ Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- ✓ Un proyecto o estructura de implementación que logra una finalidad determinada.
- ✓ Un lenguaje de programación de alto nivel.
- ✓ Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- ✓ Conexiones entre componentes de programas.
- ✓ La forma de un diagrama de objeto o de un modelo de objeto.

Para la implementación del framework se utilizó el siguiente patrón:

Patrón MVC (Modelo-Vista-Controlador):

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos.

- ✓ **Modelo:** Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o portes en un carrito de la compra.
- ✓ **Vista:** Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario
- ✓ **Controlador:** Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

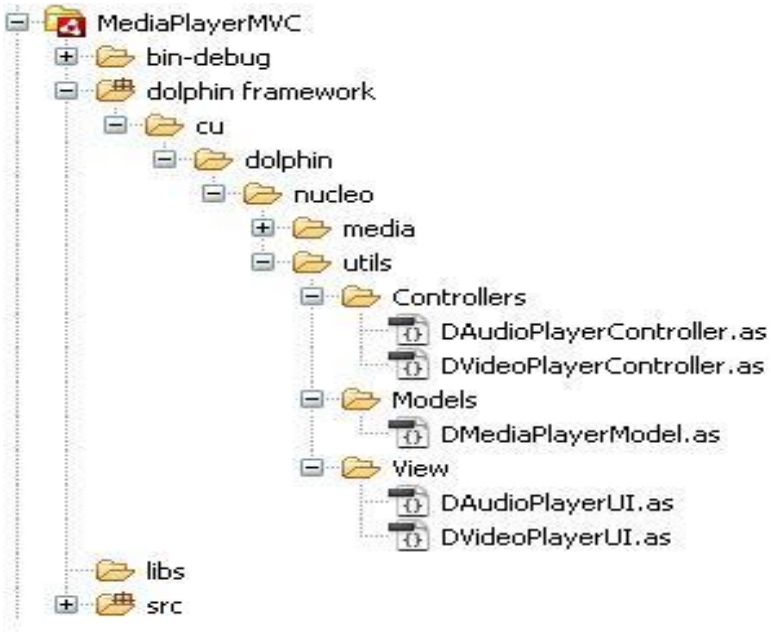

Figura 3.2.3. Estructura del patrón de diseño MVC.

3.2.4. Estándar de codificación

Los estándares de codificación, también llamados estilos de programación o convenciones de código, no son más que convenios para escribir código fuente en ciertos lenguajes de programación. Estos estándares elevan la mantenibilidad del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo, entre otras cosas, mucho más eficiente.

Para la implementación del framework de Dolphin, se decidió utilizar el estándar Adobe Open Source Coding Conventions (Convenciones de Código Fuente Abierto de Adobe). Este estándar es totalmente consistente con el estándar ECMAScript.

Algunas de las pautas que establece el estándar escogido son:

Comentarios: Compatible ASDoc, una utilidad que genera la referencia html de los comentarios que hemos puesto en el código.

Los comentarios de ASDoc empiezan por /** y acaban por */ (en Flex Builder los distinguirán porque se vuelven azules en lugar del acostumbrado verde), y necesariamente van justo antes de la definición en el código de una clase, propiedad, método. En ellos podemos escribir código html normal, con sus para negrita, párrafos encerrados en etiquetas , ya que aunque saltemos de línea en el comentario de flex, en el html se verá como una línea y entidades html cuando tengamos que usar símbolos como < (&lt) o @ (&#64, ya que se usa para poner tags).

Comentarios MXML

```
<!---
     Comentarios a nivel de clase.
 --->
```

Comentarios AS 3.0: Cada nueva línea del comentario en AS3 empieza por un asterisco y un espacio (para mayor legibilidad).

Después de la descripción, se pueden poner tags que cumplen diferentes funciones. Algunos son estos:

@**param**- Describe el parámetro de un método.

```
/**
 * Este método es el encargado de graficar un ejercicio determinado
 * devolviendo el objeto de IU que se mostrara
 *
 * @paramp_ejercicio    Ejercicio que se va a graficar
 * @paramp_tipo         Tipología de ejercicio seleccionada
 */
```

Declaraciones:

Constantes: Los nombres de las constantes deben escribirse con mayúsculas en su totalidad y se deben separar las palabras con underscore.

```
var CANTIDAD_ESTUDIANTES: Number = 10;
```

Variables: Los nombres de las variables deben escribirse con palabras compuestas sin espacios y con la primera letra de la segunda palabra en mayúscula.

```
var conexionesActuales: Number = 0;
```

Sentencias for: Las sentencias que se encuentren dentro de un ciclo, deben colocarse siempre dentro de llaves, incluso si es una sola línea de código. Si la variable límite del ciclo es una función que no necesita ser reevaluada en cada iteración, debe crearse una variable llamada n justo encima del ciclo e igualarse a esta función. Siempre se debe dejar un espacio entre la palabra reservada **"for"** y el paréntesis izquierdo, al igual que entre los elementos del ciclo y los operadores que se encuentran dentro del mismo.

```
for (var i:Number=0; i < n; i++)
    {
        /** Implementación */
    }
```

En el caso de los ciclos anidados la variable de control del ciclo exterior se debe llamar **i** y a la variable límite **n**, mientras que en el ciclo interior se deben nombrar **j** y **m** respectivamente.

Sentencias while: Las líneas de código que se encuentran dentro de una sentencia de este tipo deben ser encerradas entre llaves.

```
while (i < n)
      {
       /**Implementación*/
      }
```

Siempre se debe dejar un espacio entre la palabra reservada **"while"** y el paréntesis izquierdo, al igual que entre los operadores que se encuentran dentro de la sentencia.

Clases: Los nombres de las clases deben escribirse con palabras compuestas sin espacios y con la primera letra de cada palabra en mayúscula.

```
public class MiClase extends MovieClip
```

Interfaces: Los nombres de las interfaces deben escribirse con una **"I"** antes de las palabras compuestas, sin espacios y con la primera letra de cada palabra en mayúscula.

```
interface IConectorBaseDatos
```

Métodos: Los nombres de los métodos deben escribirse con palabras compuestas sin espacios y con la primera letra de la segunda palabra en mayúscula.

```
private function actualizarObservadores():void
```

3.3. Descripción del XML

Como se expresó anteriormente el lenguaje XML fue el escogido para almacenar la información ofrecida por la aplicación, debido a las ventajas que brinda el mismo y a las facilidades que ofrece para el trabajo con la herramienta Adobe Flex Builder 3. Para el desarrollo del sistema propuesto se confeccionó un estándar XML para cada uno de los contenidos que conforman los cursos educativos de la plataforma educativa Dolphin.

XML de contenidos

```
<container height="" width="" id="" direction="">
   <Text id="" text="" fontWeight="" fontFamily="" fontSize="">
   </Text>
</container>
```

XML de imágenes

```
<container height="" width="" id="" direction="">
  <Image width="" height="" source="" id=""/>
</container>
```

XML de sonidos

```
<container width="" horizontalGap="" y="">
  <Canvas width="" height="">
    <Button label=">" id="playBtn" x="" width=""height=""/>
    <Button label="||" id="pauseBtn" width="" height=""/>
  </Canvas>
  <Canvas width="" height="">
    <ProgressBar id="loadProgress" label="" minimum="" maximum=""/>
    <HSlider id="posicionSld" x="-" y="" height="" visible=""/>
  </Canvas>
  <Canvas width="" height="" x="" y="">
    <Label id="duracionLb" x="" y="" width="" text="00:00"/>
    <container id="spectrum" width="35” height="" x="" y=""/>
    <HSlider id="volumenSld" width="50" height="20" value=""/>
  </Canvas>
</container>
```

XML de videos

```
<container width="" horizontalGap="" y="">
   <container width="100%" minHeight="100" height="280">
      <VideoDisplay id="videoDisplay" autoPlay="false"/>
   </container>
   <Canvas width="" height="">
      <Button label=">" id="playBtn" x="" width=""height=""/>
      <Button label="||" id="pauseBtn" width="" height=""/>
   </Canvas>
   <Canvas width="" height="">
     <ProgressBar id="loadProgress" label="" minimum="" maximum=""/>
     <HSlider id="posicionSld" x="-" y="" height="" visible=""/>
   </Canvas>
   <Canvas width="" height="" x="" y="">
     <Label id="duracionLb" x="" y="" width="" text="00:00"/>
     <container id="spectrum" width="35” height="" x="" y=""/>
     <HSlider id="volumenSld" width="50" height="20" value=""/>
   </Canvas>
</container>
```

XML de actividades interactivas

```
 <container height="" width="" id="" direction="">
  <Text id="" text="" fontWeight="" fontFamily="" fontSize="" >
    <texto><![CDATA[]]></texto>
  </Text>
 </container>
<inciso>
  <container height="" width="" id="" direction="">
    <Image width="" height="" source="" id=""/>
  </container>
  <respuesta height="" width="" id="" direction="horizontal">
    <container height="" id="" direction="vertical">
      <CheckBox selected="" id="" alpha="" enabled="" width=""/>
    </container>
    <container height="" id="" direction="vertical">
      <Text id="" text="" fontWeight="" fontFamily="" fontSize=""/>
    </container>
  </respuesta>
</inciso>
```

3.4. Desarrollo de las iteraciones

En la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar, teniendo en cuenta las necesidades requeridas por el cliente. Durante el transcurso de las iteraciones se lleva a cabo una revisión del plan de iteraciones y se modifica en caso de ser necesario. Como parte de este plan, se descomponen las HU en tareas de programación o ingeniería, asignando a un equipo de desarrollo (o una persona), responsable de su implementación, aplicando la práctica de la programación en parejas. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores.

Teniendo en cuenta la planificación realizada con anterioridad, se llevó a cabo el desarrollo del sistema en tres iteraciones, obteniéndose como finalidad un producto con todas las restricciones y características deseadas por el cliente. A continuación se detallan cada una de las iteraciones.

3.4.1. Iteración 1

En esta iteración se le da cumplimiento a la implementación de las HU que corresponden a los números 1 y 2, consideradas de mayor importancia para el desarrollo del framework, con el fin de obtener una versión del producto con algunas de las funcionalidades críticas como: generar un XML para cada una de las medias que conforman los cursos de la plataforma educativa Dolphin y correspondientemente cargar dichos archivos XML.

Tabla 3.4.1 HU abordadas en la primera iteración

Historias de Usuario	**Tiempo de Implementación (semanas)**	
	Estimación	**Real**
Generación de XML	1	1
Carga de XML	1	1

Tabla 3.4.1.1ª Tarea de la HU #1

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Crear un estándar XML para cada una de las medias (texto, imágenes, audio, video, actividades interactivas de selección).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/03/2010	**Fecha fin:** 23/03/2010
Programador responsable: Pedro Luis Rojas Lemus.	
Descripción: Pretende solucionar el problema de expresar información estructurada de la manera más abstracta y reutilizable posible.	

Tabla 3.4.1.1ª Tarea de la HU #2

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 2
Nombre de la tarea: Cargar XML	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23/03/2010	**Fecha fin:** 30/03/2010
Programador responsable: Pedro Luis Rojas Lemus.	
Descripción: Esta tarea se realiza para agilizar el proceso de carga de la plataforma educativa Dolphin y así evitar que se demore en visualizar cualquier contenido o información que el usuario desee ver dentro de la plataforma.	

3.4.2. Iteración 2

En esta iteración se implementaron las HU que corresponden a los números 3 y 4. Dichas HU son las que brindan las funcionalidades de reproducción de audio y video a través de una galería de sonidos y videos.

Tabla 3.4.2 HU abordadas en la segunda iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Reproductor de Audio.	1	1
Reproductor de Video.	1	1

Tabla 3.4.2.1ª Tarea de la HU #3

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 3
Nombre de la tarea: Crear plantilla de Reproductor de Sonido y Contenidos (textos e imágenes).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 30/03/2010	**Fecha fin:** 5/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Consiste en la realización del diseño del reproductor de audio.	

Tabla 3.4.2.2ª Tarea de la HU #3

Tarea de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 3
Nombre de la tarea: Creación del Reproductor de Audio.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 30/03/2010	**Fecha fin:** 5/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Implementación de las clases que complementan el reproductor de audio.	

Tabla 3.4.2.3ª Tarea de la HU #4

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 4
Nombre de la tarea: Crear plantilla de Reproductor de Video y Contenidos (textos e imágenes).	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 5/04/2010	**Fecha fin:** 12/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Consiste en la realización del diseño del reproductor de video.	

Tabla 3.4.2.4ª Tarea de la HU #4

Tarea de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 4
Nombre de la tarea: Creación del Reproductor de Video.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 5/04/2010	**Fecha fin:** 12/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Implementación de las clases que complementan el reproductor de video.	

3.4.3. Iteración 3

En esta última iteración del framework, se le dio cumplimiento a la HU 5. Dicha HU desempeña la funcionalidad de crear actividades interactivas y validar las respuestas de las mismas.

Tabla 3.4.3 HU abordadas en la tercera iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Actividad Interactiva de Selección.	1	1

Tabla 3.4.3.1ª Tarea de la HU #5

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 5
Nombre de la tarea: Crear plantilla de Actividad Interactiva Selección Simple.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/04/2010	**Fecha fin:** 19/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Consiste en la realización del diseño de los componentes que componen la actividad interactiva selección simple.	

Tabla 3.4.3.2ª Tarea de la HU #5

Tarea de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 5
Nombre de la tarea: Creación de Actividad Interactiva Selección Simple.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/04/2010	**Fecha fin:** 19/04/2010
Programador responsable: Pedro Luis Rojas Lemus	
Descripción: Implementación de las clases que componen la actividad interactiva selección simple.	

Tabla 3.4.3.3ª Tarea de la HU #5

Tarea de Ingeniería	
No. De la Tarea: 3	**No. De la HU:** 5
Nombre de la tarea: Crear plantilla de Actividad Interactiva Selección Múltiple.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 19/04/2010	**Fecha fin:** 26/04/2010
Programador responsable: Leynier Viquillon Lavorí.	
Descripción: Consiste en la realización del diseño de los componentes que componen la actividad interactiva selección múltiple.	

Tabla 3.4.3.4ª Tarea de la HU #5

Tarea de Ingeniería	
No. De la Tarea: 4	**No. De la HU:** 5
Nombre de la tarea: Creación de Actividad Interactiva Selección Múltiple.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 19/04/2010	**Fecha fin:** 26/04/2010
Programador responsable: Pedro Luis Rojas Lemus.	
Descripción: Implementación de las clases que componen la actividad interactiva selección múltiple.	

3.5. Pruebas

Uno de los pilares de la metodología XP es el uso de las pruebas para comprobar el funcionamiento de los códigos que se vayan implementando. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones (24).

La metodología ágil XP divide las pruebas en dos grupos: pruebas unitarias y pruebas de aceptación. Las pruebas unitarias son desarrolladas por los programadores y se encargan de verificar el código automáticamente y las pruebas de aceptación están destinadas a verificar que al final de cada iteración las Historias de Usuario cumplen con la funcionalidad asignada y satisfagan las necesidades del cliente.

Las pruebas de aceptación son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente, por esto el cliente es la persona adecuada para diseñar las pruebas de aceptación.

3.5.1. Desarrollo dirigido por pruebas

El desarrollo dirigido por pruebas (TDD), es una práctica de programación que involucra otras dos prácticas: Escribir las pruebas primero (Test First Development) y Refactorización (Refactoring). Para escribir las pruebas generalmente se utiliza la Prueba Unitaria (unit test en inglés).

Primeramente se escribe una prueba y se verifica que las pruebas fallen, luego se implementa el código que haga que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito. El propósito del desarrollo guiado por pruebas es lograr un código limpio que funcione (Del inglés: Clean code that works). La idea es que los requerimientos sean traducidos a pruebas, de este modo, cuando las pruebas pasen se garantizará que los requerimientos se hayan implementado correctamente.

TDD permite un diseño más robusto, tanto es así que a menudo se piensa TDD como Diseño manejado por las pruebas (Test Driven Design). En consecuencia, TDD facilita un diseño más mantenible a través de la noción de pruebas. Estas pruebas obligan a reflexionar sobre el comportamiento del código y la forma de garantizar que funciona según lo previsto. La mayoría de las veces, el código influenciado por TDD es relativamente seguro, y sin duda, es bastante simple (25).

La práctica de TDD puede resumirse en 3 simples reglas:

- ✓ No se puede escribir código productivo, a menos que sea para hacer pasar un test fallido.
- ✓ No se puede escribir más que lo necesario para que falle un test unitario; los errores de compilación se consideran fallos.
- ✓ No se puede escribir más código productivo del estrictamente necesario para hacer pasar un test (25).

Ciclo de desarrollo TDD

1. **Escribir la prueba**. Para escribir la prueba, el desarrollador debe entender claramente las especificaciones y los requisitos. El diseño del documento deberá cubrir todos los escenarios de prueba y condición de excepciones.
2. **Escribir el código haciendo que pase la prueba**. Este paso fuerza al programador a tomar la perspectiva de un cliente considerando el código a través de sus interfaces. Ésta es la parte conducida por el diseño, del TDD. Como parte de la calibración de la prueba, el código debe fallar la prueba significativamente las primeras veces.
3. **Ejecutar las pruebas automatizadas**. Si pasan, el programador puede garantizar que el código resuelve los casos de prueba escritos. Si hay fallos, el código no resolvió los casos de prueba.
4. **Refactorización y limpieza en el código**. Después se vuelven a efectuar los casos de prueba y se observan los resultados.

5. **Repetición**. Después se repetirá el ciclo y se comenzará a agregar las funcionalidades adicionales o a arreglar cualquier error (25).

Debido a que ActionScript no cuenta con un framework que permita realizar las pruebas unitarias o el TDD, se hace necesario utilizar el depurador de Adobe Flex Builder 3 o la consola de salida en última instancia.

3.5.2. Pruebas de aceptación

Las pruebas de aceptación son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

Las pruebas de aceptación son consideradas como “pruebas de caja negra”. Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos. Así mismo, en caso de que fallen varias pruebas, deben indicar el orden de prioridad de resolución. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación.

Dado que la responsabilidad es grupal, es recomendable publicar los resultados de las pruebas de aceptación, de manera que todo el equipo esté al tanto de esta información.

Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

- ✓ Identificar todas las acciones en la historia de usuario.
- ✓ Para cada acción escribir al menos una prueba.
- ✓ Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.
- ✓ Para otros datos, reemplazar las entradas que hacen que la acción falle, y registrar los resultados.

Tabla 3.5.2.1 Prueba de aceptación para la HU “Generación de XML”

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Comprobar la generación de XML.	
Descripción: Evaluar la generación de XML de cada uno de los componentes que conforman las medias de la plataforma educativa Dolphin.	
Condiciones de Ejecución: El cliente debe probar que se generan correctamente todos los XML.	
Entrada/ Pasos de ejecución: Probar la generación de cada uno de los XML existentes.	
Resultado Esperado: Se generan correctamente los XML.	
Evaluación de la Prueba: -	

Tabla 3.5.2.2 Prueba de aceptación para la HU “Carga de XML”

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 2
Nombre: Comprobar la carga de XML.	
Descripción: Evaluar la carga de cada uno de los componentes que conforman las medias de la plataforma educativa Dolphin.	
Condiciones de Ejecución: El cliente debe probar que se cargan correctamente todas las medias que conforman la plataforma educativa Dolphin.	
Entrada/ Pasos de ejecución: Probar la carga y lectura de los XML existentes.	
Resultado Esperado: Se cargan correctamente los XML.	
Evaluación de la Prueba: -	

Tabla 3.5.2.3 Prueba de aceptación para la HU “Reproductor de Audio”

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** 3
Nombre: Reproductor de Audio	
Descripción: Evalúa las funcionalidades del reproductor de audio.	
Condiciones de Ejecución: El cliente debe probar todas las funcionalidades del reproductor.	
Entrada/ Pasos de ejecución: Todas las funcionalidades del reproductor son probadas.	
Resultado Esperado: Se reproducen correctamente los sonidos, archivos *.mp3.	
Evaluación de la Prueba: -	

Tabla 3.5.2.4 Prueba de aceptación para la HU “Reproductor de Video”

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** 4
Nombre: Reproductor de Video	
Descripción: Evalúa las funcionalidades del reproductor de video.	
Condiciones de Ejecución: El cliente debe probar todas las funcionalidades del reproductor.	
Entrada/ Pasos de ejecución: Todas las funcionalidades del reproductor son probadas.	
Resultado Esperado: Se reproducen correctamente los videos, archivos *.flv.	
Evaluación de la Prueba: -	

Tabla 3.5.2.5 Prueba de aceptación para la HU “Actividad Interactiva de Selección”

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** 5
Nombre: Actividad Interactiva de Selección Simple.	
Descripción: Prueba para la funcionalidad de las actividad interactiva de selección simple.	
Condiciones de Ejecución: El cliente debe probar que todas las actividades interactivas de selección simple requeridas, cumplan con la expectativa esperada.	
Entrada/ Pasos de ejecución: Se solicita la actividad interactiva de selección simple deseada y posteriormente se procede a probar que dicha actividad funcione correctamente.	
Resultado Esperado: Las actividades interactivas de selección simple no presentan errores.	
Evaluación de la Prueba: -	

Tabla 3.5.2.6 Prueba de aceptación para la HU “Actividad Interactiva de Selección”

Caso de Prueba de Aceptación	
Código: HU5_P2	**Historia de Usuario:** 5
Nombre: Actividad Interactiva de Selección Múltiple.	
Descripción: Prueba para la funcionalidad de las actividad interactiva de selección múltiple.	
Condiciones de Ejecución: El cliente debe probar que todas las actividades interactivas de selección múltiple requeridas, cumplan con la expectativa esperada.	
Entrada/ Pasos de ejecución: Se solicita la actividad interactiva de selección múltiple deseada y posteriormente se procede a probar que dicha actividad funcione correctamente.	
Resultado Esperado: Las actividades interactivas de selección múltiple no presentan errores.	
Evaluación de la Prueba: -	

3.6. Conclusiones

En este capítulo se construyó el diagrama de clases del framework, logrando una visión detallada de sus atributos y las relaciones entre ellas. Se elaboran las tarjetas CRC y se definió un estándar de codificación a utilizar. Se desarrollaron las tareas correspondientes para dar solución a las historias de usuarios y se definió el modelo arquitectónico a seguir. Se realizaron las pruebas unitarias y se definieron las pruebas de aceptación. Con la finalización de este capítulo se da por terminada la propuesta de solución de la aplicación a implementar por el grupo de desarrollo.

Conclusiones generales

El presente trabajo se enfocó en el estudio y desarrollo de un framework para la generación automática de interfaces gráficas de los contenidos que conforman los cursos educativos de la plataforma Dolphin, a partir del trabajo con XML.

Como resultado del trabajo realizado se logró el diseño e implementación de un framework con las siguientes ventajas:

- ✓ Ahorro en tiempo y recursos para las personas que utilicen la plataforma educativa Dolphin.
- ✓ Fácil mantenimiento, de modo que se le puedan adicionar nuevos módulos a la plataforma.
- ✓ Facilidad de uso debido al desarrollo de un conjunto de plantillas o contenidos prediseñados.
- ✓ Reutilización de código y componentes en proyectos con características similares.

El sistema cuenta con la documentación necesaria para su entendimiento, mantenimiento y posterior escalabilidad debido al uso de la metodología de desarrollo ágil XP, que permitió construir los artefactos requeridos. Al terminar el trabajo se concluyó que es factible crear un framework que permita reutilizar código y componentes del diseño, constituyendo un paso significante en la producción de plataformas educativas.

Recomendaciones

Debido a los resultados de la investigación efectuada y de la experiencia adquirida durante la realización de este trabajo, y con el propósito de asegurar la posterior ampliación, modificación y mejora del framework propuesto, se exponen a continuación algunas recomendaciones:

- ✓ Mantener actualizado el framework propuesto, incorporándole las recomendaciones realizadas por los expertos.
- ✓ Continuar con el proceso de desarrollo del framework con el objetivo de implementar nuevas tipologías de ejercicios de selección e incorporarlas a las plantillas existentes.
- ✓ Implementar estándares XMLs compatibles con otras plataformas educativas existentes; por ejemplo Moodle.

Referencias bibliográficas

1. **Iriarte, Leandro.** http://issuu.com/azunino/docs/iriarte-tesis. [En línea] 2008.

2. **Lacleta, Pablo López García y María Luisa Sein-Echaluce.** Moodle: difusión y funcionalidades. [En línea] http://profesores.universia.es/seccionEspecial.jsp?idEspecial=12&idSeccion=4230&title=CARACTERISTICAS-MOODLE.

3. Aureba Servicios Web. [En línea] http://www.aurebaweb.com/diseno-web/e-learning/moodle.

4. Conociendo el programa Hot Potatoes. [En línea] http://www.educar.org/enlared/misquiz/hotpotatoes.htm.

5. La revolución pedagógica: el entorno Moodle . [En línea] http://profesores.universia.es/seccionEspecial.jsp?idEspecial=12&idSeccion=4232&title=OTRAS-PLATAFORMAS-PEDAGOGICAS.

6. **Ibarra, Livan Kabir Badías.** *Propuesta de Procedimiento de Evaluación de los Frameworks. Un enfoque práctico.* Ciudad de la Habana, : s.n., 2007.

7. *eumed.net.* [Online] Grupo de investigación eumednet (SEJ-309) de la Universidad de Málaga, 2009. http://www.eumed.net.

8. **José H. Canós, Patricio Letelier yMª Carmen Penadés.** Métodologías Ágiles en el Desarrollo de Software. [En línea] http://www.willydev.net/descargas/prev/TodoAgil.pdf.

9. **Morón, Universidad .** Metodologías Ágiles. [En línea] http://noqualityinside.com/nqi/nqifiles/Metodologias_Agiles.pdf.

10. **M., Itzcoalt Alvarez.** Desarrollo Ágil con SCRUM. [En línea] http://www.sg.com.mx/sg07/presentaciones/Mejora%20de%20procesos/SG07.P02.Scrum.pdf

11. **Calderon, Anyelin.** Sistematizando Aprendizajes . [En línea] http://anyelincalderon.blogspot.com/2010/04/algo-sobre-metodologias-agiles.html.

12. **Marlon Rojas Güemes, Boris Luis Álvarez Enq.** *Software Educativo como soporte tecnológico del aprendizaje técnico-táctico del Futsal para los estudiantes de la Universidad de las Ciencias Informáticas.* 2009.

13. **informaticos, Departamento de lenguaje y Sistemas.** *Introducción a Rational Rose.* Barcelona : s.n.

14. Visual Paradigm for UML (CE) [Mac OS X]. [En línea] 2007. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(Iglesia_Anglicana)_%5BMac_OS_X_cuenta_14717_p/.

15. **Vargas, Yoennis Garrido.** *Arquitectura para la creación de aplicaciones multimedia. MAPri.* Ciudad de la Habana : s.n., 2009.

16. *Páginas personales en la web.* [Online] http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html.

17. Actionscript 3.0 : Más potencia para Flash. *Ciberaula.* [Online] 2006. http://flash.ciberaula.com/noticia/as3/.

18. *Primavera.* [Online] Universidad de las Ciencias Informáticas, 2008. http://primavera.uci.cu.

19. *aplicaciones empresariales.com.* [Online] Agosto 16, 2008. http://www.aplicacionesempresariales.com/adobe-flex-3-un-manejador-de-aplicaciones-open-source.html.

20. Ficha de Artículo. *multimediateam.* [Online] http://www.mmteamglobal.com/fichaProducto.asp?cod=98577.

21. **Morales, Sergio.** http://carloscar7.files.wordpress.com/2008/02/carlos-david-carballo-espana.doc. [En línea]

22. **Casanovas, Joseph.** http://www.willydev.net/descargas/UsabilidadArquitectura.pdf. [En línea]

23. **Ramírez., Lic. Elisa Arizaca.** http://virtual.usalesiana.edu.bo/web/practica/archiv/compon.doc. [En línea] 2009.

24. **J. J. Gutiérrez, M. J. Escalona, M. Mejías, J. Torres.** *PRUEBAS DEL SISTEMA EN PROGRAMACIÓN.* University of Sevilla : s.n.

25. Dos Ideas. *http://www.dosideas.com/wiki/Test_Driven_Development.* [En línea]

Bibliografía

Martínez, Prof. Lucy Córdova. Actividades de evaluación del Curso: El Aprendizaje Cooperativo y las TIC. [En línea] 2008.

Conociendo el programa Hot Potatoes. [En línea] http://www.educar.org/enlared/misquiz/hotpotatoes.htm.

Roberth G. Figueroa, Camilo J. Solís. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. [En línea].

BOLIVARIANA, UNIVERSIDAD UNION. PROGRAMACIÓN EXTREMA (XP). [En línea] http://www.mex.tl/images/18149/PROGRAMACI%C3%93N%20EXTREMA.pdf.

Llanes, Daniel Sánchez. "SGIEPC. SISTEMA DE GESTIÓN DE INFORMACIÓN EN LA EMPRESA PROCESADORA DE CAFÉ ELADIO MACHÍN". [En línea] http://www.monografias.com/trabajos-pdf/sgiepc/sgiepc.pdf.

Lenguajes de Programación Orientada a Objetos . [En línea] lunes 18 de mayo de 2009. http://www.blogterrier.com.ar/2009/05/lenguajes-de-programacion-orientada.html.

Avello, Daniel Gayo. ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA EN INFORMÁTICA DE OVIEDO. [En línea] http://danielgm.es/des/PFC.pdf.

Vargas, Yoennis Garrido. *Arquitectura para la creación de aplicaciones multimedia. MAPri.* Ciudad de la Habana : s.n., 2009.

Glosario de términos

GUI Graphical User Interface

J2EE Java Platform, Enterprise Edition o Java EE

PDA Personal Digital Assistant

ECMAScript: Especificación de lenguaje de programación publicada por ECMA International. Su desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262

Software: Equipamiento lógico o soporte lógico de un ordenador. Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Suele ser utilizado para referirse a los programas y aplicaciones informáticas.

Tecnologías de la Información y las Comunicaciones: conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.

Software Educativo: Software destinando a la enseñanza y el auto aprendizaje y que además permite el desarrollo de ciertas habilidades cognitivas.

Plataformas educativas: software que permite estimular la idea de cooperación y de interacción, como aspectos centrales del proceso de aprendizaje y enseñanza, mediante el uso de herramientas colaborativas que favorecen la adquisición de aprendizajes significativos en los estudiantes y que al mismo tiempo afianzan en los docentes prácticas de enseñanza mediadas por las Tecnologías de la Información y la Comunicación (TIC).

Metodología: Métodos de investigación que se siguen para alcanzar una gama de objetivos en una ciencia.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Código abierto: (en inglés open source) es el término con el que se conoce al software distribuido y desarrollado libremente.

HTML: es el lenguaje de marcado predominante para la construcción de páginas web.

Generación de código: es una de las fases mediante el cual un compilador convierte un programa sintácticamente correcto en una serie de instrucciones a ser interpretadas por una máquina.

Licencia Pública General de GNU: más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Lenguaje de Programación: Conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

Interfaz de Programación de Aplicaciones (API): Conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.