Universidad de las Ciencias Informáticas

Facultad 8

Diseño Investigación Científica.

“Análisis y diseño de la herramienta para desarrollar los ejercicios de autoevaluación en Libros Electrónicos.”

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas.

Autor:

Yeney González Chong.

Tutor:

Ing**.** Abduly Díaz García

“Ciudad de la Habana. Julio, 2008”

“Año 50 de la Revolución”

Declaración de Autoría

Yo Yeney González Chong, como única autora de la presente tesis, reconozco 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_____.

_____

Firma del Autor

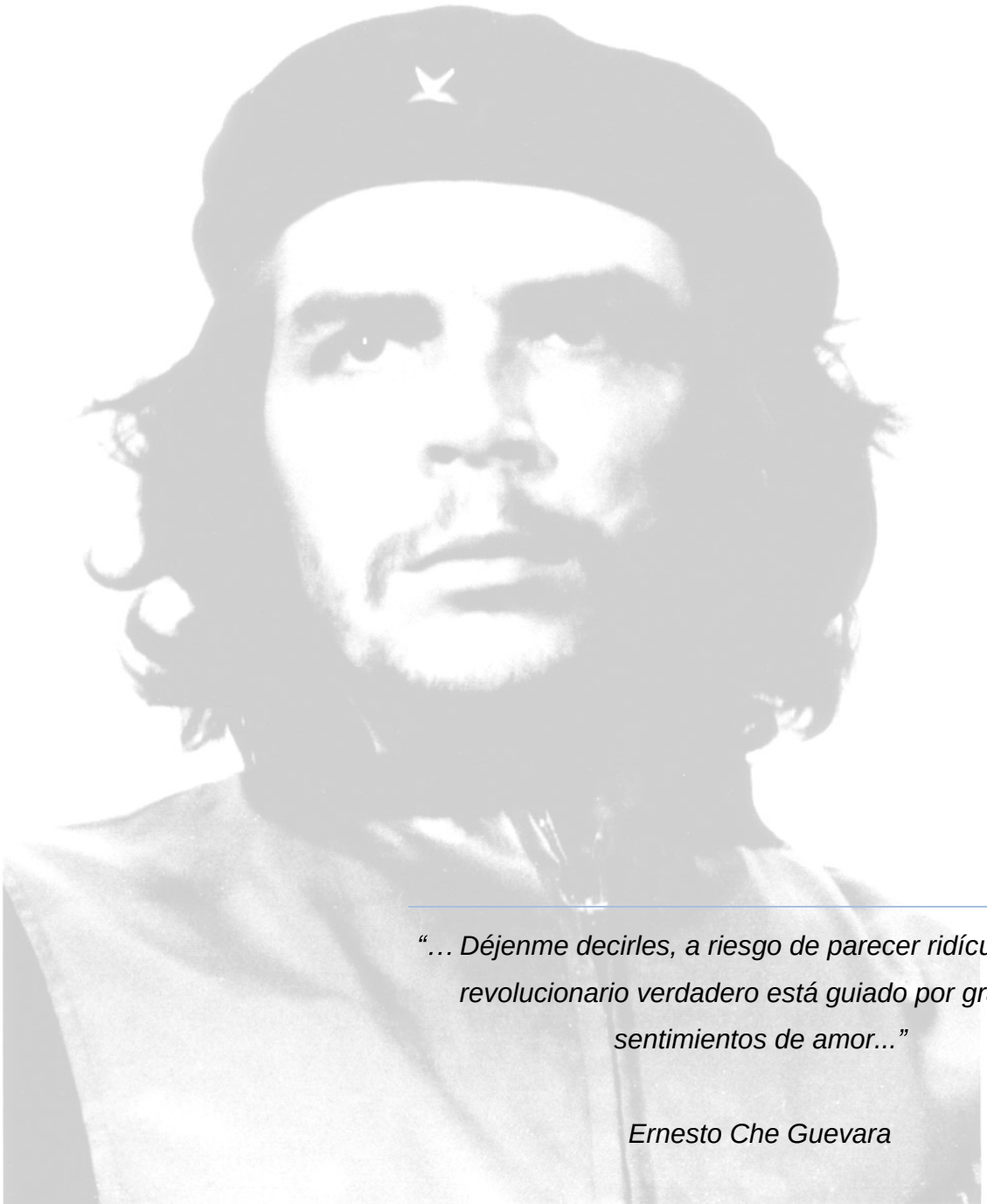
"Yeney González Chong”

_____

Firma del Tutor

“Abduly Díaz García”

Pensamiento



"… Déjenme decirles, a riesgo de parecer ridículo, que el revolucionario verdadero está guiado por grandes sentimientos de amor..."

Ernesto Che Guevara

Agradecimientos

Primeramente le agradezco a mi familia en especial a mis padres por haberme brindado todo su apoyo desde mi ingreso a esta universidad, especialmente en esta última etapa que es tan importante para mí; además del apoyo de mi hermano y demás familiares que aunque no intervinieron directamente influenciaron en mi desarrollo durante toda mi etapa universitaria.

Le agradezco a mi tutor Abduly Díaz García, que aparte de guiarme en este trabajo de diploma y soportar todos mis retrasos, me brindo su apoyo en la realización de este trabajo para que se realizara lo mejor posible.

A Yadiel Ramos Rodríguez, Michel Arias Arias, Damir Góngora Mora, Alden Hernández Gomez y Aliuska Garcia Machado, que me brindaron su confianza, su ayuda incondicional y sus consejos cuando los necesite.

A Yulieski Pérez Chacón, que siendo un ex compañero de tesis por causas académicas, aportó ideas y desarrolló junto conmigo parte del documento tesis.

A mis compañeros del aula que me estuvieron aguantando durante cinco años, como mis amistades que aunque no las mencione particularmente por ser muchas, me han brindado su mano y ayuda en cada instante.

A todas las personas en generar que de un modo u otro han hecho de mí quien soy.

Y el ultimo y no el menos importante, brindarle agradecimiento y mi apoyo incondicional a nuestro comandante Fidel Castro Ruz, por esta gran oportunidad de convertirme en un ser útil en nuestra sociedad.

Dedicatoria

"A mis padres, familiares y amigos que siempre me apoyaron y me llenaron de felicidad en mi vida".

Resumen

En el marco de la batalla de ideas en que se encuentra inmerso nuestro país, la dirección de la Revolución puso en marcha uno de los proyectos educacionales más grandes desarrollado en los últimos tiempos, la Universidad de las Ciencias Informáticas (UCI), una institución de nuevo tipo capaz de formar ingenieros informáticos con el fin de informatizar y desarrollar la sociedad cubana. Con el crecer de la comunidad universitaria, áreas como la de producción han sido favorecidas con la incorporación de los estudiantes a ella, propiciando de esta forma un estrecho vínculo estudio/producción el cual ha sido respaldado con la introducción de numerosos proyectos tanto nacionales como extranjeros. Actualmente la facultad 8 posee dentro de su plan de producción algunos proyectos relacionados con Libros Electrónicos , los cuales han sido montado sobre plataformas Web y soportados en herramientas para la realización de autoevaluación , la cual no satisface todos los requisitos para la realización de un mejor trabajo por lo que se hace necesario confeccionar una herramientas capaz de gestionar script para autoevaluaciones con mejores funcionalidades, realizada en cualquier tecnología libre y que cumpla con las expectativas tanto de los clientes como de los desarrolladores del proyecto.

Índice

Introducción

Actualmente nuestro país se encuentra inmerso en numerosos procesos y transformaciones en las áreas de la cultura, la ciencia y la educación por lo que Proyectos de la Revolución o de la Batalla de Ideas como los ha definido nuestro comandante, se implementan diariamente con el único objetivo de desarrollar una cultura general integral en nuestro pueblo y apoyar la economía del país. Es así como surge la Universidad de las Ciencias Informáticas (UCI), cuya responsabilidad se enmarca en el desarrollo e informatización de una sociedad en alza dentro del campo de las tecnologías de la información y que ha comenzado a aportar con sus ya terminados productos o software en numerosos campos de la sociedad.

La educación y salud son unos de los campos de la sociedad que hoy en día transforman sus metodologías de estudios, integrando dentro de sus programas, las Tecnologías de Información y Comunicaciones (TIC) con diversos software o recursos informáticos que hacen aun más didácticas las actividades docentes y que posibilitan que estas puedan extraditarse al personal médico y educativo que se encuentra fuera de nuestro país. Muchos de estos recursos como los libros electrónicos son implementados en proyectos productivos de la facultad 8 de la UCI y no son más que la digitalización dinámica de contenidos educativos de todas las esferas de la educación de la salud, entregados en copia dura o CD para su posterior distribución. Muchos de estos productos digitales están destinados no solo a la transmisión de conocimientos sino a la comprobación de los mismos, por lo que son acompañados de numerosos ejercicios para la autoevaluación del aprendizaje de cada unidad de estudio, por lo que se hace necesario para los desarrolladores el dominio de herramientas informáticas que puedan automatizar el desarrollo de dichas autoevaluaciones y que estas puedan ser insertadas dentro de los libros electrónicos según los criterios de los clientes .

Frecuentemente, las aplicaciones realizadas con la herramienta que posee en uso el proyecto de Libros Electrónico para el desarrollo de las mismas, en esta facultad, han estado acompañadas de múltiples deficiencias debido a la generación de un script que se torna difícil en cuanto a su interpretación, localización de errores y manejo de los desarrolladores. Esta consta de una aplicación Web separada en dos pequeños módulos, uno para la generación de los script de las preguntas y otro para la comprobación y ejecución de los mismos. La herramienta presenta algunas funcionalidades no terminadas dentro de su estructura, posee una arquitectura pobre en diseño además de no contar con ayuda alguna que posibilite algún tipo de información al desarrollador y no consta como un producto realizado por la universidad en aspectos legales.

Debido a todo lo citado, se ha planteado como **situación problémica** la necesidad de la realización del análisis y el diseño de una herramienta informática capaz de crear las autoevaluaciones de los libros electrónicos que se producen en la facultad 8, debido a que la herramienta actual no es la idónea y no facilita la comprensión adecuada para su uso por parte de los desarrolladores, y con la puesta en práctica de esta nueva versión se podría facilitar un mejor entendimiento a la hora de producirlo.

Este trabajo surge para dar respuesta a las deficiencias existentes en la facultad 8, relacionados con la herramienta basada en la realización de autoevaluaciones en los libros electrónicos en los proyectos de la facultad 8, por lo que el **problema a solucionar** en él, consiste en:

¿Cómo desarrollar el análisis y diseño de una nueva aplicación para automatizar a la confección de las autoevaluaciones en la producción de libros electrónicos que actualmente se implementan en la facultad 8?

Por tanto el **objeto de estudio** es el análisis y el diseño de la herramienta para generar objetos de aprendizaje.

De ello se deriva que el **campo de acción** que abarca este trabajo, es el análisis y el diseño de la herramienta para generar autoevaluaciones de los libros electrónicos que se producen en la facultad 8.

Idea a defender

Si se realiza el análisis y diseño de una herramienta que automatice la implementación de las autoevaluaciones; se piensa que será posible alcanzar:

- Una herramienta más fácil de manejar.
- Una interfaz más atractiva para el desarrollador.
- Mayores funcionalidades.
- Un mejor entendimiento entre los desarrolladores y los clientes, logrando un resultado que satisfaga todas las necesidades de los usuarios.

El **objetivo general** de la investigación es la realización del análisis y el diseño de una herramienta que permita generar las autoevaluaciones en el proyecto de los libros electrónicos.

Como **objetivos específicos** se plantean los siguientes:

- Realizar un estudio de investigación sobre la creación de las autoevaluaciones de libros electrónicos en los proyectos que se dedican a su desarrollo.
- Realizar el análisis y diseño, de un sistema que permita generar las autoevaluaciones.
- Crear un documento que recoja todo el proceso investigativo del desarrollo del sistema informático.

Para poder dar cumplimiento, de una forma completa y exitosa, a estos objetivos se ha decidido desarrollar las siguientes **tareas**:

- Elaborar el diseño teórico de la investigación.
- Determinar los requisitos que debe cumplir el sistema para su culminación exitosa.
- Revisar estado del arte.
- Investigar sobre las herramientas y tecnologías a utilizar.
- Elaborar la propuesta de solución.
- Analizar y diseñar la herramienta.
- Estudio de la factibilidad de dicho sistema.
- Estructurar el documento tesis.

Aportes prácticos esperados del trabajo

Con este trabajo se espera que se cumplan las siguientes expectativas:

- Disminuir el tiempo en la realización de las autoevaluaciones.
- Un mejor diseño en la versión final de la herramienta.
- Presentar una ayuda a los desarrolladores.

Este documento consta de 3 capítulos donde la información está distribuida de la siguiente forma:

Capítulo 1. Fundamentación Teórica: Se muestran las materias teóricas necesarias para la visión del trabajo y la comprensión de las posibles herramientas a utilizar.

Capítulo 2. Modelo de Dominio, Requerimientos y Descripción Sistema:
Se definirán todos los procesos, actores, trabajadores y casos de uso del negocio; a partir de los diagramas de modelo del dominio.

Capítulo 3. Análisis y Diseño del Sistema: Diagramas de análisis y diseño, además de una descripción de la arquitectura , patrones utilizados , estándares de interfaz y un formato de los reportes, también una breve descripción de la ayuda y el tratamiento de errores en el sistema.

Capítulo 1 Fundamentación Teórica.

1.1 Introducción

En este capítulo se describirá la situación correspondiente a las Autoevaluaciones de los libros electrónicos ante la situación problémica planteada y se hará una exploración de las herramientas para la generación de autoevaluaciones existentes relacionadas con el campo de acción propuesto. Se analizara las tendencias y tecnologías actuales sobre las cuales se apoya la propuesta brindada.

1.2 Situación en los Proyectos Productivos

Actualmente en el proyecto de Libros Electrónicos de la facultad 8 posee insuficiencias respecto a la generación de autoevaluaciones para la culminación de las tareas o ejercicios didácticos de cada producto electrónico debido a que la herramienta con que se cuenta no satisface los requisitos necesarios para su elaboración.

La aplicación actual además de poseer las insuficiencias mencionadas anteriormente no permite la realización de un solo check (chequeo de las respuestas aceptadas para cada ejercicio realizado) de rectificación una vez terminados todos los ejercicios, sino que para cada ejercicio se realiza en su continuación un chequeo evaluativo y donde en múltiples ocasiones la intención del cliente es que exista un solo check de comprobación al final de cada estructura para que de esta forma el evaluado se vea en la necesidad de realizar todos los ejercicios sin acudir a una búsqueda inmediata de los resultados.

Son pocas las ventajas que brinda dicha herramienta, por lo que se hace necesaria la búsqueda de una nueva versión que facilite el trabajo de los desarrolladores de proyecto en cuanto a la muestra de información o ayuda, rectificación de errores y mejor interfaz , para de esta forma facilitar los procesos productivos y la satisfacción de los clientes.

1.3 Existencia en Cuba y el Mundo

Después de navegar en Internet se han efectuado varias investigaciones que presentan la información del sistema que se quiere gestionar, además de encontrar algunos datos que aunque no presentan suficiente información detallada del proceso. Presenta una visión cercana de lo que se quiere alcanzar. Ejemplo:

1.3.1 Informe de Autoevaluaciones 2006 (CentroGeo)

Durante el 2006, CentroGeo, ha seguido con su proceso de consolidación y de resultados, en particular, en aspectos relativos al programa de postgrado; la formalización de conocimiento; la realización de proyectos; y el fortalecimiento de vínculos a nivel nacional e internacional que refuerzan su posicionamiento entre la comunidad científica en sus áreas de especialidad. Por lo que es el deseo de la Dirección General del CentroGeo además de dar cumplimiento a los ordenamientos institucionales en materia de auto evaluación, continuar con la valiosa comunicación con los miembros del Órgano de Gobierno y, compartir los éxitos logrados como institución, como profesionistas y como personas.[1]

1.3.2 Welcome to English - Net.

Es una presentación Web, donde se espera el fortalecimiento del ingles, haciendo uso de diferentes actividades, en donde podemos encontrar la realización de autoevaluaciones: En este proyecto las actividades de evaluación permitirán al capacitando evaluar su aprendizaje, lo cual servirá para interpretar la marcha de su aprendizaje.

Ayudando también a la Evaluación a distancia: siendo el sitio un lugar donde los capacitados pueden participar libremente la evaluación será de carácter formativa y no obligatoria. [2]

1.3.3 Estándares para un mundo moderno.

La Preparación de los Estudiantes para el Futuro.

Este articulo informa y explica la importancia que tiene para este siglo la Alfabetización digital, siendo las auto evaluaciones un medio para medir tanto el contenido como las habilidades que ayudarán a preparar a los estudiantes; por lo que puede ofrecer el aprendizaje basado en tecnología que antes no era posible, potenciar el proceso de aprendizaje o simplemente enriquecer el diseño de las lecciones con los avances en el entendimiento logrados por otros educadores profesionales.[3]

1.3.4 El libro electrónico: una alternativa para la mejora de la calidad en la Educación Superior.

El diseño y elaboración de libros-e (Libros electrónicos) se perfila como una de las herramientas pedagógicas de apoyo para sus estudios, desarrollándose mediante aplicaciones de la Web y multimedia, o sea, es una mejor práctica en al auto-estudio, mediante la realización de la autoevaluación, proporcionando grandes beneficios al estudiantado como:

1. Homogeneización de los conocimientos utilizando el Sistema Maestro, que con ayuda del generador de ejercicios, los estudiantes tendrán a su disposición ejercicios con diferentes grados de complejidad, entre los que pueden elegir hasta que consideren haber comprendido el material. (Límite de comprensión de habilidad) y antes de que su tarea sea evaluada por el Sistema Maestro.

2. El aprendiz desarrollará habilidades de investigación así como propuestas Independientes del uso de las herramientas, utilizadas en las respectivas tareas. El aprendiz, tendrá la posibilidad de generar sus propias propuestas para ser incluidas en el libro electrónico, con fines de fomentar la iniciativa, la motivación intrínseca y la necesidad de alcanzar mayor complejidad en sus destrezas y conocimientos, favoreciendo en el corto y largo plazo, mayor autonomía en su aprendizaje. Es decir, el aprendiz podrá aplicar las herramientas y la estructura aprendida y ejercitada durante el curso, para desarrollar sus propias propuestas y páginas del libro electrónico.

3. Interactividad. Este concepto ha generado polémica en este tipo de propuestas debido a la falta de socialización de los individuos involucrados en la dinámica. En este punto se ha diseñado un sistema de interactividad entre los estudiantes que estén trabajando al mismo tiempo, para que puedan intercambiar sus dudas e información.[4]

El sistema maestro

Como se mencionó anteriormente, el sistema maestro se presenta como una herramienta útil para el proceso y para el estudiante, ya que esta diseñado para proveer de ejercicios, casos a los estudiantes, en los que se promueve la comprensión de los modelos a nivel teórico y práctico. Además de mediatiza la información y apoya al estudiante en la conceptualización, en la comprensión de los conceptos y temas cruciales del curso.

1.3.5 La autoevaluación de la carrera, una vía para el mejoramiento de la calidad académica.

En América Latina, incluyendo Cuba, se observa un incremento cuantitativo de los procesos de evaluación y la calidad se ha convertido en tema de preocupación por parte de las instituciones. La experiencia mundial combina mecanismos de autoevaluación y de evaluación externa, en la casi totalidad de sistemas de

acreditación, aunque se reconoce que la autoevaluación es una de las formas más adecuadas para asegurar el avance constante hacia una mayor calidad.

El objetivo general del Sistema Evaluación y Acreditación de Carreras Universitarias es la elevación de la calidad del proceso de formación en las carreras universitarias, por lo que se constituye en una herramienta fundamental para la gestión del mejoramiento continuo de la calidad en la formación de los profesionales.

Etapas del Proceso de Autoevaluación:

Primera etapa**:** Es una fase preparatoria y debe incluir actividades.

Segunda Etapa**:** La característica de esta etapa es la recopilación de información, a partir de indicadores seleccionados previamente, en relación con las dimensiones a estudiar.

Tercera etapa: Se caracteriza por la interpretación y valoración con un carácter integrador**,** contextual e histórico de la información obtenida, según criterios delimitados previamente para cada variable. Se elabora el informe de autoevaluación, que responde a un diagnóstico de la realidad de la carrera con la identificación de fortalezas y debilidades en la variable que les corresponde.

Cuarta etapa: Consiste en la producción de recomendaciones y la propuesta de soluciones para el mejoramiento, sostenido y derivado lógicamente de las fortalezas y debilidades identificadas previamente.

Quinta etapa: Seguimiento de la implementación del plan de acción.

1.3.5 Herramientas para la producción de materiales didácticos para las modalidades de enseñanza semipresencial y a distancia.

Representa una breve presentación del escenario y las herramientas para la producción de materiales didácticos para las modalidades de enseñanza semipresencial y a distancia en las universidades en Cuba.

El desarrollo de nuevos materiales didácticos sobre la base de diversos software para complementar el diseño curricular de cada disciplina, asignatura o curso puede sintetizarse como por ejemplo: La evaluación o autodiagnóstico por medio del paquete integrado *Hot Potatoes*.

Además exhibe como una polémica entre los docentes la evaluación entre los docentes, más cuando se trata de formación en línea o educación a distancia; es por eso que al usar la plataforma como medio de enseñanza, el claustro universitario cubano ha diseñado estrategias de evaluación, dirigidas hacia el desarrollo del autoaprendizaje en el estudiante mediante una correcta autorregulación cognitiva en el proceso formativo, un control que puede seguir constantemente la búsqueda de errores y el desarrollo de las habilidades metacognitivas del estudiante. *Hot Potatoes*, una herramienta de evaluación incluida en los módulos del paquete *Moodle*, es una elección positiva para desarrollar procesos de autoevaluación, control y regulación del aprendizaje de los estudiantes.[5]

1.3.7 Herramienta para el desarrollo de ejercicios para los libros electrónicos.

La Universidad de las Ciencias Informáticas, tiene entre sus líneas de investigaciones, el desarrollo de asignaturas o montajes de cursos virtuales sobre plataforma *Moodle*, seleccionada por la institución en el año 2006 por sus

facilidades tecnológicas y pedagógicas para desarrollar el proceso de enseñanza-aprendizaje en ambientes virtuales. Un ejemplo es la generación de autoevaluaciones del proyecto Libros Electrónicos; aunque la herramienta que se utiliza presenta errores, en donde le daremos solución con una nueva versión mejorada.

1.4 Tendencias y tecnologías actuales.

1.4.1 Libros electrónicos.

También conocido como libro digital o eBook, el libro electrónico es una publicación cuyo soporte no es el papel sino un archivo electrónico, su texto se presenta en formato digital y se almacena en diskette, CD-ROM o en Internet. El libro electrónico permite incorporar elementos multimedia como vídeo, audio, y en el caso de Internet, posibilita enlaces a otras páginas de libros digitales de la red. [6]

1.4.2 Herramienta Informática.

Herramienta:

Aplicación empleada para la construcción (de ahí su nombre) de otros programas o aplicaciones informáticas.[7]

1.4.2.1 Herramientas Utilizadas

Dreamweaver

Es la herramienta de diseño de páginas Web más avanzada, tal como se ha afirmado en muchos medios. Aunque sea un experto programador de HTML el

usuario que lo maneje, siempre se encontrarán en este programa razones para utilizarlo, sobretodo en lo que a productividad se refiere.

Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar:

- Hojas de estilo y capas.
- Java script para crear efectos e interactividades.
- Inserción de archivos multimedia.

Además es un programa que se puede actualizar con componentes, que fabrica tanto Macromedia como otras compañías, para realizar otras acciones más avanzadas.

1.4.2.2 Herramienta Informática de Prueba:

Grupo de herramientas que permite reproducir la funcionalidad de una Aplicación informática mediante el uso de guiones o "scripts", tanto en la interfaz gráfica de usuario como en la comunicación de la aplicación con otras, como puede ser entre una aplicación que se ejecuta en un navegador y el servidor Web que le atiende y entre éste y una base de datos.

Internet Explorer

Es un navegador Web producido por Microsoft para el sistema operativo Windows y más tarde para Apple Macintosh y Solaris Unix. Actualmente es el navegador de Internet más popular y más utilizado en el mundo, rebasando en gran medida a las competencias existentes.

Mozilla

Es un navegador de Internet, con interfaz gráfica de usuario desarrollado por la Corporación Mozilla y un gran número de voluntarios externos . Comenzó como un derivado del Mozilla Application Suite, que terminó por reemplazarlo como el producto bandera del proyecto Mozilla, bajo la dirección de la Fundación Mozilla. El programa es multiplataforma y está disponible en versiones para Microsoft Windows, Mac OS X y GNU/Linux. Incluye un sistema propio de extensiones, que pueden instalarse por sus usuarios para personalizar el aspecto y comportamiento del navegador.

Java Script

Java Script es un lenguaje de *scripts* desarrollado por *Netscape* para incrementar las funcionalidades del lenguaje HTML. Se utiliza embebido en el código HTML, entre las tags <script> y </script>. Sus características más importantes son:

- Java Script es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias Java Script contenidas en una página HTML y ejecutarlas adecuadamente.
- Java Script es un lenguaje orientado a eventos. Cuando un usuario pincha sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante Java Script se pueden desarrollar Scripts que ejecuten acciones en respuesta a estos eventos.
- Java Script es un lenguaje orientado a objetos. El modelo de objetos de Java Script está reducido y simplificado, pero incluye los elementos necesarios para que los Scripts puedan acceder a la información de una página y puedan actuar sobre la interfaz del navegador. [8]

1.5 Lenguajes de programación para el desarrollo de Herramientas.

1.5.1 Lenguaje de programación C# (C Sharp).

Este nuevo lenguaje orientado a objetos con énfasis en internet se basa en las lecciones aprendidas de los lenguajes C, C++, Java y Visual Basic. Por ello se trata de un lenguaje que combina todas las cualidades que se pueden esperar de un lenguaje moderno (orientación a objetos y gestión automática de memoria) a la vez que proporciona un gran rendimiento.

Surgió en junio de 2000, cuando Microsoft despejó todas las dudas liberando la especificación de un nuevo lenguaje llamado C#. A esto le siguió rápidamente la primera versión de prueba del entorno de desarrollo estándar (SDK) .NET, que incluía un compilador de C#. El nuevo lenguaje estaba diseñado por Anders Hejlsberg (creador de Turbo Pascal y arquitecto de Delphi), Scott Wiltamuth y Peter Golde. Entonces describieron el lenguaje como "simple, moderno, orientado a objetos, de tipado seguro y con una fuerte herencia de C/C++".

C# es un lenguaje orientado a objetos simple, elegante y con seguridad en el tratamiento de tipos, que permite a los programadores de aplicaciones empresariales crear una gran variedad de aplicaciones. También proporciona la capacidad de generar componentes de sistema duraderos en virtud de las siguientes características:

- Total compatibilidad entre COM y plataforma para integración de código existente.
- Gran robustez, gracias a la recolección de elementos no utilizados (liberación de memoria) y a la seguridad en el tratamiento de tipos.
- Seguridad implementada por medio de mecanismos de confianza intrínsecos del código.
- Plena compatibilidad con conceptos de metadatos extensibles.

Además, es posible interaccionar con otros lenguajes, entre plataformas distintas, y con datos heredados, en virtud de las siguientes características:

- Plena interoperabilidad por medio de los servicios de COM+ 1.0 y .NET Framework con un acceso limitado basado en bibliotecas.

- Compatibilidad con XML para interacción con componentes basados en tecnología Web.
- Capacidad de control de versiones para facilitar la administración y la implementación.[9]

1.5.2 Lenguaje de programación C++.

El **C++** (pronunciado *"ce más más"* o *"ce plus plus"*) es un lenguaje de programación, diseñado a mediados de los años 1980, por Bjarne Stroustrup, como extensión del lenguaje de programación C.

Se puede decir que C++ es un lenguaje que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. Actualmente existe un estándar, denominado ISO C++, al que se han adherido la mayoría de los fabricantes de compiladores más modernos. Existen también algunos intérpretes como ROOT (enlace externo).

Las principales características del C++ son:

- Programación orientada a objetos: La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real. Además, permite la reutilización del código de una manera más lógica y productiva y para el uso de plantillas o programación genérica (*templates*).
- Portabilidad: Un código escrito en C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios.
- Brevedad: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobretodo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- Programación modular: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++

con código producido en otros lenguajes de programación como Ensamblador o el propio C

- Velocidad: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel y a la reducida medida del lenguaje.

Además posee una serie de propiedades difíciles de encontrar en otros lenguajes de alto nivel:

- Posibilidad de redefinir los operadores (sobrecarga de operadores)
- Identificación de tipos en tiempo de ejecución (*RTTI*)

C++ está considerado por muchos como el lenguaje más potente, debido a que permite trabajar tanto a alto como a bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae (obliga a hacerlo casi todo manualmente al igual que C) lo que "dificulta" mucho su aprendizaje.

1.5.3 Lenguaje de programación Java.

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).
Sus principales características son:

Lenguaje simple

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo escribir applets interesantes desde el principio. Todos aquellos familiarizados con C++ encontrarán que Java es más sencillo, ya que se han eliminado ciertas características, como los punteros. Debido a su semejanza con C y C++, y dado que la mayoría de la gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

Orientado a objetos

Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos se agrupan en estructuras encapsuladas, tanto sus datos como los métodos (o funciones) que manipulan esos datos. La tendencia del futuro, a la que Java se suma, apunta hacia la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente a la arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows NT, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñada para transportar el código eficientemente a múltiples plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas. Estas dos últimas características se conocen como la Máquina Virtual Java (JVM).

Alto rendimiento: Multihebra

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

Produce applets

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java. Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc. [10]

1.5.4 Selección de Lenguaje a Utilizar.

Se utilizará Visual C# 2005 Express Edition por ser un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo y su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK.* Aunque posee patentes en la Microsoft, C# es el mas indicado para la creación de la herramienta a desarrollar debido a que posee compatibilidad con XML para interacción con componentes basados en tecnología Web y que son de usos necesarios para la generación de las autoevaluaciones que luego serán usadas en los Libros Electrónicos . Por se gran utilidad dentro de los procesos docentes de la facultad 8, posemos afirmar que actualmente es el lenguaje de programación mas utilizado en el desarrollo de herramientas y aplicaciones de desktop, por lo que se impone en cuanto a recursos, bibliografía, instaladores y personal de apoyo para la facilitación de ayuda y corrección de posibles errores.[9]

1.6 Metodologías de desarrollo de software.

La Metodología de Desarrollo de Software es el resultado de más de 15 años de experiencia en el desarrollo de sistemas, tiempo en el que se han “cosechado” las mejores prácticas, producto de haber gestionado numerosos proyectos de gran envergadura. Entre sus características se encuentra:

- Está embebida en todo el proceso de desarrollo.
- Abarca a todos los roles involucrados.
- Es flexible, permite manejar grandes proyectos como pequeños requerimientos.
- Está controlada en todas las fases, es decir “SE CUMPLE”.
- No posee pasos teóricos que nadie aplica.
- Permite ser monitoreada.

Para dar una idea de qué metodología podemos utilizar y cual se adapta más al medio, se mencionan tres de ellas que se consideran las más importantes, tal como: RUP, XP y MSF.

1.6.1 Rational Unified Process (RUP)

La metodología RUP, llamada, se divide en 4 fases de desarrollo:

- **Inicio**: el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración**: en esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción**: en esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transmisión**: el objetivo es llegar a obtener el release del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de Desarrollo

- Ingeniería de Negocios: Entendiendo las necesidades del negocio. Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
- Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte

- Configuración y administración del cambio: Guardando todas las versiones del proyecto.
- Administrando el proyecto: Administrando horarios y recursos.
- Ambiente: Administrando el ambiente de desarrollo.
- Distribución: Hacer todo lo necesario para la salida del proyecto

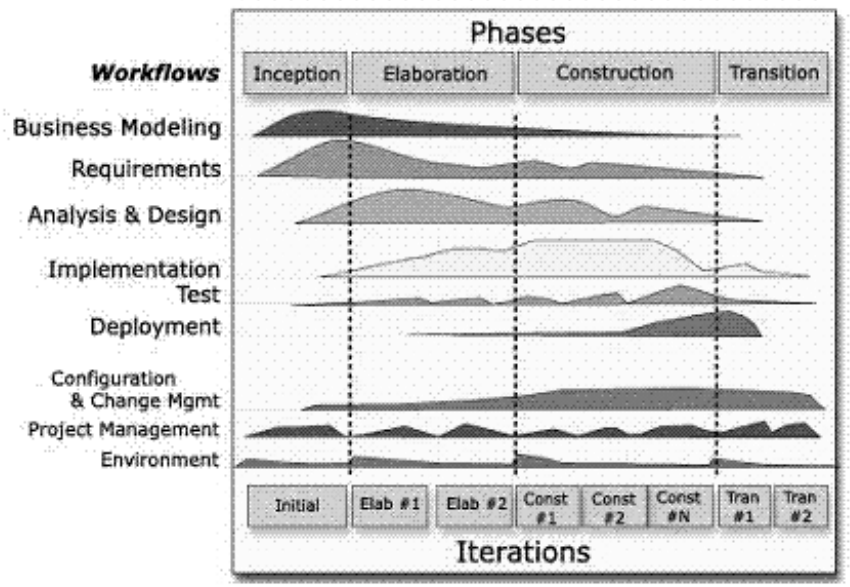


Fig. 1 Fases e Iteraciones de la Metodología RUP

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración. Los elementos del RUP son:

- Actividades: son los procesos que se llegan a determinar en cada iteración.
- Trabajadores: vienen hacer las personas o entes involucrados en cada proceso.
- Artefactos: un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

1.6.2 Extreme Programing (XP)

La Programación Extrema, o eXtreme Programming, es otra de las metodologías de desarrollo de software que existen en la actualidad. Mientras que el RUP intenta reducir la complejidad del software por medio de estructura y la preparación de las tareas pendientes en función de los objetivos de la fase y actividad actual, XP, como toda metodología ágil, lo intenta por medio de un trabajo orientado directamente al objetivo, basado en las relaciones interpersonales y la velocidad de reacción.

Supone la disposición en todo momento, por parte del equipo de trabajo, de un representante competente del cliente, que debe estar en condiciones de dar una

respuesta rápida y correcta a cualquier pregunta del equipo de desarrollo de forma que no se retrase la toma de decisiones.

La metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, podamos hacer pruebas de las fallas que pudieran ocurrir. Es como si nos adelantáramos a obtener los posibles errores.

- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio.

- Programación en pares: una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. Es como el chofer y el copiloto: mientras uno conduce, el otro consulta el mapa.

La base para el desarrollo del software que usa esta metodología son las llamadas User Stories, historias escritas por el cliente en las que describe escenarios sobre el funcionamiento del sistema y que no sólo están limitados a la interfaz de usuario, sino que también pueden describir modelos, dominio, etc. Estas User Stories junto a la arquitectura que se persigue, sirve de base para crear un plan de “entregas de software” entre el equipo de desarrollo y el cliente, para cada una las cuales se definen objetivos y las iteraciones (generalmente cortas) necesarias para cumplirlos. Las User Stories y los casos de pruebas son la base sobre la que se asienta el trabajo del desarrollador. Esta metodología apuesta por iteraciones cortas que generan software que el cliente puede ver.

Que propone XP

- Empieza en pequeño y añade funcionalidad con retroalimentación continua.
- El manejo del cambio se convierte en parte sustantiva del proceso.
- El costo del cambio no depende de la fase o etapa.
- No introduce funcionalidades antes que sean necesarias.
- El cliente o el usuario se convierte en miembro del equipo.

Derechos del Cliente

- Decidir que se implementa
- Saber el estado real y el progreso del proyecto
- Añadir, cambiar o quitar requerimientos en cualquier momento
- Obtener lo máximo de cada semana de trabajo
- Obtener un sistema funcionando cada 3 o 4 meses

Derechos del Desarrollador

- Decidir como se implementan los procesos
- Crear el sistema con la mejor calidad posible
- Pedir al cliente en cualquier momento aclaraciones de los requerimientos
- Estimar el esfuerzo para implementar el sistema
- Cambiar los requerimientos en base a nuevos descubrimientos

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores
- La simplicidad, al desarrollar y codificar los módulos del sistema
- La retroalimentación, concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales

1.6.3 Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

MSF tiene las siguientes características:

- Adaptable: es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- Escalable: puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- Flexible: es utilizada en el ambiente de desarrollo de cualquier cliente.
- Tecnología Agnóstica: porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- *Modelo de Arquitectura del Proyecto:* Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- *Modelo de Equipo:* Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo. Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.
- *Modelo de Proceso:* Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida

del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.

- *Modelo de Gestión del Riesgo:* Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.
- *Modelo de Diseño del Proceso:* Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- *Modelo de Aplicación:* Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

1.6.4 Feature Driven Development (FDD)

Se considera FDD que está a medio camino entre RUP y XP, aunque en realidad es más bien una metodología ligera. Está pensada para proyectos con un tiempo de desarrollo relativamente corto (menos de un año). Se basa en un proceso iterativo con iteraciones cortas de aproximadamente dos semanas que producen un software funcional, el cual puede ser examinado por el cliente y la dirección de la empresa. Cada iteración se define en término de funcionalidades (de ahí su nombre) que son pequeñas partes del sistema con significado para el cliente.

Esta metodología separa en cinco fases al proyecto, las cuales están determinadas por el: desarrollo de un modelo general, construcción de la lista de funcionalidades, plan de entregas sobre la base de las funcionalidades a

implementar, diseño basado en las funcionalidades e implementación basada en las funcionalidades. Todo el trabajo se realiza en grupo, aunque siempre hay un responsable, que generalmente tiene mayor experiencia, que dice la última palabra en caso de no llegar a un acuerdo.

Las funcionalidades de cada entrega se dividen entre los distintos subgrupos del equipo y se implementan. El código escrito (las clases) tiene propietario, o sea, sólo quien lo crea puede modificarlo, lo que no ocurre en XP. Es por eso que en un subgrupo deben estar todos los propietarios de las clases implicadas, pudiendo un desarrollador pertenecer a varios subgrupos. También se contemplan como parte del proceso de implementación, la preparación y ejecución de pruebas, las revisiones de código y la integración de las partes que componen el software.

1.7 Metodología a Utilizar

Como metodología a utilizar, se escogió RUP, debido a que es la más aceptada para los procesos de desarrollo de herramientas de gestión, además de que las demás metodologías según algunos autores son casos particulares de RUP. Otro de los aspectos fundamentales que determino la selección de esta metodología es que es la mas estudiada en la universidad, permitiendo así la existencia de mayor documentación y ayuda para su uso.

1.7.1 Rational Rose Enterprise Edition

Es una herramienta Lower CASE, que permite el diseño detallado del software y la generación de código fuente (de programas y bases de datos) e ingeniería inversa (obtención del diseño a partir del código fuente), basado en modelos con soporte UML. Es una forma de ayuda para la comprensión del sistema y de sus distintos componentes. Su característica más significativa consiste en la creación de componentes, que contengan una serie de archivos dentro de los cuales se encuentran las distintas clases pertenecientes a dicho componente.[11]

1.8 Plataforma de desarrollo y Sistema Operativo utilizado

1.8.1 Plataforma .NET

Utilizada tanto en servicios Web como aplicaciones tradicionales (aplicaciones de consola, aplicaciones de ventanas, servicios de Windows NT, etc.), publicada por la Microsoft bajo el denominado kit de desarrollo de software conocido como **.NET Framework SDK**, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución y **Visual Studio.NET**, que permite hacer todo la anterior desde una interfaz visual basada en ventanas. Ambas herramientas puede descargarse gratuitamente desde [http://www.msdn.microsoft.com/net], aunque la última sólo está disponible para subscriptores MSDN Universal (los no subscriptores pueden pedirlo desde dicha dirección y se les enviará gratis por correo ordinario).

1.8 Sistema Operativo

Se utilizara Windows XP sp2 por su factibilidad tanto con las herramientas, lenguaje y plataforma de desarrollo a utilizar, además de ser el sistema con que actualmente se trabaja dentro del proyecto de Libros Electrónicos.

1.9 Conclusiones

En el presente capítulo se ha hecho referencia al las tecnologías y tendencias actuales existentes tanto en Cuba como en el mundo referente a herramientas y software para realización de Autoevaluaciones así como una descripción de algunos de los principales conceptos tocados en la investigación y una exposición de los principales lenguajes de programación, metodologías de desarrollo y herramientas a utilizar justificando el uso de cada una de las seleccionadas en el proceso de desarrollo de la herramienta.

Capítulo 2 Características del Sistema

2.1 Introducción

En este capítulo se va a describir a describir el proceso propuesto ante la situación problémica planteada, además de enfatizar en las reglas establecidas. Al mismo tiempo se definirán los requerimientos funcionales y no funcionales con

los que debe cumplir y como modelarán los objetos o eventos más importantes que ocurren en el contexto del sistema mediante un modelo de dominio, y se describirán los conceptos asociados la aplicación al igual que el actor. Se identificaron los actores y trabajadores del mismo, así como el diagrama de casos de uso y su formato expandido. Se plantea el análisis del sistema utilizando para su modelado el Lenguaje Unificado de Modelación (UML), que permite representar el diagrama de casos de uso del sistema. Por lo que se realizó una descripción detallada de cada caso de uso con sus correspondientes requisitos funcionales y no funcionales.

2.2 Especificación del contenido.

De modo general, la herramienta cuenta con las opciones de generar página de autoevaluación y modificar autoevaluación, además en la opción de generar página de autoevaluación tenemos diferentes elecciones como:

- Incluir ejercicio de Completamiento.
- Incluir ejercicio de Selección.
- Incluir ejercicio de Verdadero y Falso.
- Incluir ejercicio de Tabular.

En la Generación de Página de Autoevaluación se cuenta con un documento que no es más que los ejercicios entregados por los clientes, como guía en el desarrollo de los ejercicios. Apoyándose en la herramienta hace las especificaciones de los ejercicios propuesto en la documentación obteniéndose un fichero (.js) que en conjunto con la biblioteca de la herramienta y el navegador donde se obtiene la página de autoevaluación.

En la Modificación de Autoevaluaciones se debe tener la solicitud de parte de los clientes que es lo que desean cambiar o cuando se haga las pruebas necesarias en el navegador después de terminada cada página se encuentre un error, se entra en el código intermedio y se cambia manualmente.

2.3 Descripción del sistema propuesto

Para la creación de páginas de autoevaluación y modificación de la misma, se creará una herramienta que generará las autoevaluaciones. Aparte de la página de autoevaluación, también creará un fichero con la información de la autoevaluación, para una posterior modificación de esta en caso que sea necesario.

2.3.1 Requerimientos.

Requerimiento no es más que una condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.

Los requerimientos o requisitos se pueden clasificar en: **funcionales y no funcionales.**

2.3.1.1 Requerimientos Funcionales

Son capacidades o condiciones que el sistema debe cumplir y se mantienen invariables sin importar con que propiedades o cualidades se relacionen.

1. Generar Página de Autoevaluación.
- 1.1 Incluir Ejercicio de Completamiento.
- 1.2 Incluir Ejercicio de Selección.
- 1.3 Incluir Ejercicio de Verdadero y Falso.
- 1.4 Incluir Ejercicio de Tabular.
- 1.5 Incluir Imagen.
- 1.6 Incluir sonido.

2. Modificar Autoevaluación.
- 2.1 Modificar Ejercicio de Completamiento.
- 2.2 Modificar Ejercicio de Selección.
- 2.3 Modificar Ejercicio de Verdadero y Falso.
- 2.4 Modificar ejercicio de Tabular.
- 2.5 Cambiar imagen.

2.6 Cambiar sonido.

3. Generar archivo para modificar la autoevaluación

4. Cargar archivo para modificar autoevaluación.

5. Mostrar vista previa de la autoevaluación que se está desarrollando.

2.3.1.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Interfaz Externa

- Debe contar con una interfaz amigable y de fácil manejo para los usuarios.
- Ajustarse a los estándares de diseño.
- Contar con una gran interactividad.

Usabilidad

- Deberá ser usado por todos los desarrolladores del proyecto, permitiendo una fácil interpretación de la misma.

Soporte

- Fácil mantenimiento y de configuración sencilla.
- Contara con una ayuda para el usuario.
- Al final de cada ejercicio se realizaran pruebas para posibles errores.

Software

- El cliente debe tener uno de los siguientes navegadores: Mozilla, Mozilla Firefox, Microsoft Internet Explorer (para Windows 5.0 o superior).

2.4 Modelo Conceptual

2.4.1 Modelo de Negocio

El modelo de negocio es un modelo que describe los procesos de un negocio y su interacción con elementos externos tales como socios y clientes, es decir,

describe las funciones que el negocio pretende realizar y su objetivo básico es describir cómo el negocio es utilizado por sus clientes y socios.

Los objetivos del modelamiento del negocio son:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema.
- Comprender los problemas actuales de la organización e identificar las mejoras potenciales.
- Asegurar que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

[12]

2.4.2 Modelo de Dominio

Representa conceptos del mundo real, no de los componentes del software. Un modelo del dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. Este tipo de modelo no incluyen las responsabilidades de las personas que ejecutan las actividades.

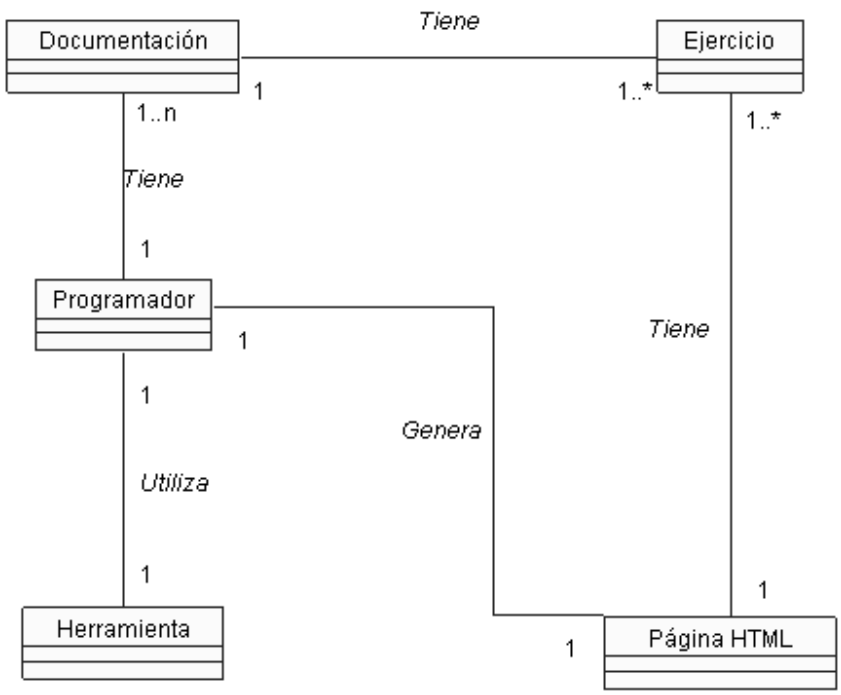
El modelo de dominio se describe mediante diagramas de clases de UML, especialmente diagramas de clases, donde se muestran y especifican las principales clases conceptúales (clases del dominio) que pueden intervenir en el sistema, y cómo se relacionan unas con otras mediante asociaciones.

2.4.3 ¿Modelo de Negocio o Modelo de Dominio?

Ya que no esta clara la frontera de este proceso de negocio se hace uso del modelado del dominio, que nos brinda una vista de los conceptos más importantes en nuestro sistema. Para mayor conocimiento se realiza un diagrama de clases para las principales clases conceptuales del proceso.

2.4.4 Modelo de Dominio

MODELO DEL DOMINIO



2.4.4.1 Conceptos del Dominio

Documentación: Es un Documento Word entregado por el cliente que contiene la información relacionados con los ejercicios que se deben generar en el proyecto de los Libro Electrónico.

Ejercicio: Pregunta para medir el conocimiento de los estudiantes basados en los contenidos dados.

Programador: Es la persona que crea las autoevaluaciones dentro del proyecto, basándose en la información entregada por el cliente en la documentación.

Herramienta: Es el programa utilizado por el programador que genera de forma automática las páginas de autoevaluaciones de los ejercicios propuestos en la documentación.

Página HTLM: Fichero creado por el programador utilizando la herramienta en la cual se representan los ejercicios ya generados

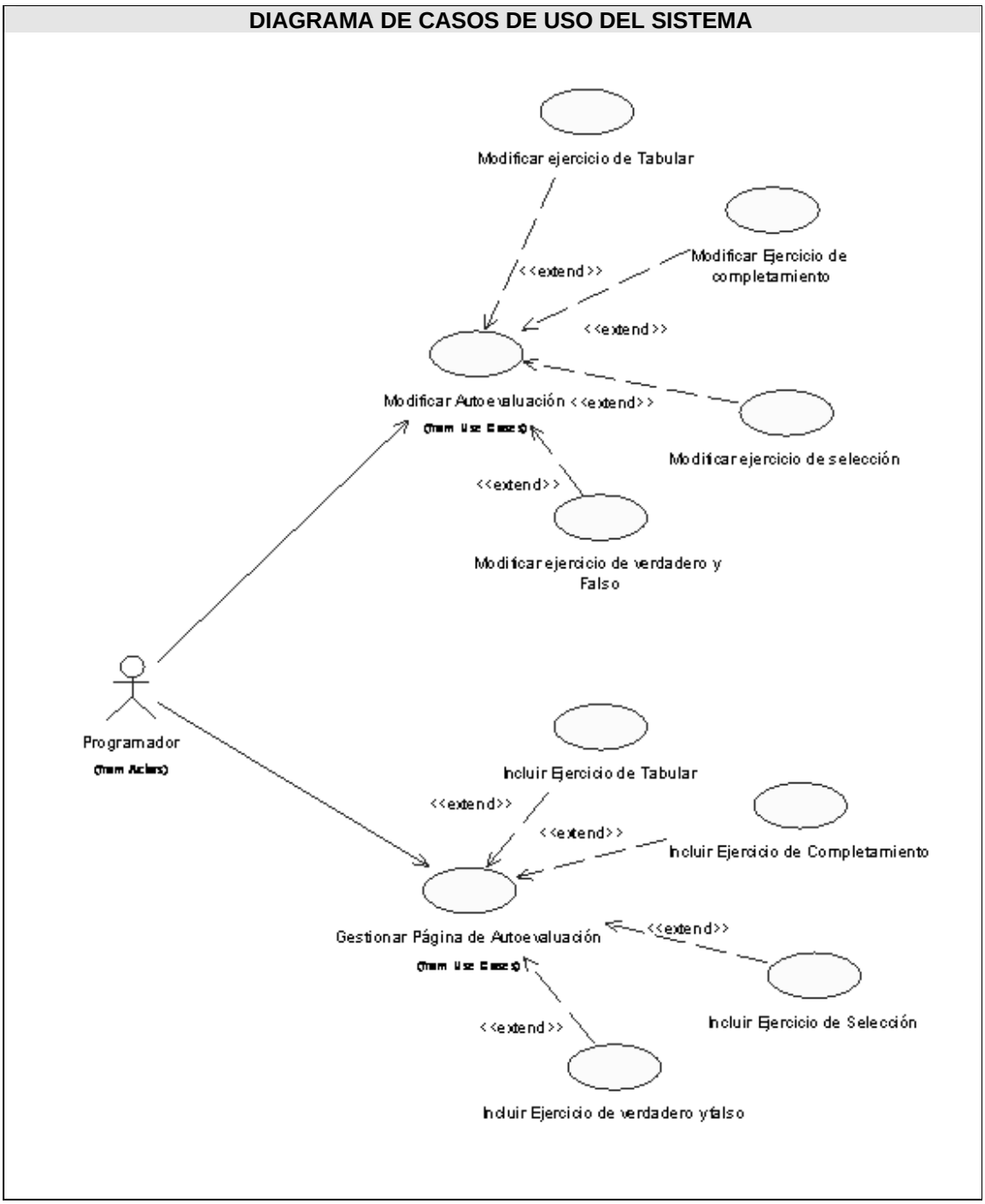
2.5 Modelo de Caso de uso del sistema

2.5.1 Definición de los Actores del Sistema

Programador	Persona que crea las autoevaluaciones dentro del proyecto

2.5.2 Diagrama de CU del Sistema

DIAGRAMA DE CASOS DE USO DEL SISTEMA



2.5.3 Descripción de los Casos de Uso del Sistema

<table>
<tr><td colspan="2">Caso de uso:</td><td colspan="2">Gestionar Página de Autoevaluación</td></tr>
<tr><td colspan="2">Actores :</td><td colspan="2">Programador</td></tr>
<tr><td colspan="2">Propósito:</td><td colspan="2">Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad</td></tr>
<tr><td colspan="4">Resumen:
El caso de uso se inicia cuando el programador abre la aplicación para elaborar una página de autoevaluación. Apoyándose en la herramienta obtiene dos fichero (.HTML) y (.XML); que en conjunto con la biblioteca de la herramienta y el navegador se puede mostrar la página de autoevaluación.</td></tr>
<tr><td colspan="4">Referencias:
RF 1</td></tr>
<tr><td colspan="4">Precondiciones:
Contener la documentación de las autoevaluaciones.</td></tr>
<tr><td colspan="4">Poscondiciones:
Se obtiene dos ficheros (.HTML) y (.XML).</td></tr>
<tr><td colspan="4">Curso normal de los eventos:</td></tr>
<tr><td colspan="2">Acción del actor:</td><td colspan="2">Respuesta del proceso del Sistema:</td></tr>
<tr><td>1</td><td>Inicia la aplicación</td><td>1.1</td><td>Entra a la herramienta, para la creación de un ejercicio mostrando la opción de:
-Gestionar Página de Autoevaluación (Ver Escenario Gestionar Página de Autoevaluación).
-Modificar Autoevaluación (Ver Escenario Modificar</td></tr>
</table>

			Autoevaluación).
Escenario Gestionar Página de Autoevaluación			
1	El programador selecciona la opción de Gestionar Página de Autoevaluación.	1.1	El sistema muestra la opción de : -Insertar Ejercicio de completamiento(Ver CU Crear Ejercicio de completamiento) -Insertar ejercicio de Selección.(Ver CU Crear Ejercicio de Selección) -Insertar ejercicio de Verdadero y Falso (Ver CU Crear Ejercicio de Verdadero y Falso). -Insertar Ejercicio de Tabular (Ver CU Crear Ejercicio de Tabular)
Escenario de Modificar Autoevaluación			
1	El programador selecciona la opción de Modificar Autoevaluación.	1.1	Muestra un diálogo para cargar un fichero (.XML)
2	Buscar el archivo (.XML) que tiene la información de la autoevaluación que va a modificar y oprime el botón abrir.	2.1	Muestra la vista previa de la Página de Autoevaluación y muestra las opciones de: -Modificar Ejercicio de completamiento(Ver CU Modificar Ejercicio de completamiento) -Modificar ejercicio de Selección.(Ver CU Modificar Ejercicio de Selección) -Modificar ejercicio de Verdadero y

			Falso (Ver CU Modificar Ejercicio de Verdadero y Falso). -Modificar Ejercicio de Tabular (Ver CU Modificar Ejercicio de Tabular)

Caso de Uso:		**Incluir Ejercicios de Completamiento**	
Actores :	Programador		
Propósito:	Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la creación de un ejercicio, especifica el ejercicio que desea realizar, seleccionando la opción de Completamiento.			
Referencias: RF 1			
Precondiciones: Se debe de tener bien definido el tipo de ejercicio que se quiere realizar.			
Poscondiciones: La página contiene un ejercicio de completamiento.			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere realizar.	1.1	Muestra los campos que se necesitan para confeccionar un ejercicio de completamiento.
2	El programador introduce las	2.1	El sistema muestra una vista

	especificaciones las oraciones a completar y el valor de las palabras omitidas y oprime el botón insertar pregunta.		previa del ejercicio insertado.

Caso de uso:		**Incluir Ejercicios de Verdadero y Falso**	
Actores :	Programador		
Propósito:	Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad.		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la creación de un ejercicio, especifica el ejercicio que desea realizar, seleccionando la opción de Verdadero y Falso.			
Referencias: RF 1			
Precondiciones:			
Poscondiciones: La página contiene un ejercicio de Verdadero y Falso			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere desarrollar	1.1	Muestra campos necesarios para desarrollar ejercicios de verdaderos y falsos
2	El Programador introduce diferentes respuestas, las cuales se le asigna un valor según sea	2.1	El sistema muestra una vista previa del ejercicio insertado

	verdadero o falso y oprime el botón insertar pregunta.		

Caso de uso:		**Incluir Ejercicios de Selección**	
Actores :	Programador		
Propósito:	Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad.		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la creación de un ejercicio, especifica el ejercicio que desea realizar, seleccionando la opción de Selección Sencilla.			
Referencias: RF 1			
Precondiciones: Según la documentación se debe de tener bien definido el tipo de ejercicio que se quiere realizar.			
Poscondiciones: La página contiene un ejercicio de Selección Sencilla.			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere desarrollar	1.1	El sistema muestra campos que se utilizan para la confección de ejercicios de selección sencilla.
2	El Programador introduce diferentes respuestas donde solo una de ellas la correcta o mas de una según el ejercicio que sea y oprime el botón	2.1	El sistema muestra una vista previa del ejercicio insertado

	insertar pregunta.		

Caso de Uso:		**Modificar Ejercicios de Completamiento**	
Actores :	Programador		
Propósito:	Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la modificación de un ejercicio, especifica el ejercicio con la respuesta que desea modificar.			
Referencias: RF 2			
Precondiciones: Se debe de tener bien definido el tipo de ejercicio que se quiere modificar.			
Poscondiciones: El ejercicio rectificado.			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el ejercicio que quiere modificar	1.1	El sistema muestra los campos necesarios para la modificación del ejercicio
2	El programador realiza los cambios rectificando los errores tanto en la pregunta como en las respuestas dando valores nuevos según el error y oprime el botón modificar.	2.1	El sistema muestra una vista previa del ejercicio modificado.

Caso de uso:		**Modificar Ejercicios de Verdadero y Falso**	
Actores :	Programador		
Propósito:	Crear las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad.		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la modificación de un ejercicio, seleccionando el ejercicio con su respuesta, donde se le realizan los cambios.			
Referencias: RF 2			
Precondiciones: Se debe de tener bien definido el tipo de ejercicio que se quiere modificar.			
Poscondiciones: El ejercicio rectificado			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere modificar	1.1	El sistema muestra campos necesarios para la modificación de ejercicios de verdaderos y falsos.
2	El programador realiza los cambios rectificando los errores tanto en la pregunta como en las respuestas dando valores nuevos según el error y oprime el botón modificar.	2.1	El sistema muestra una vista previa del ejercicio modificado.

Caso de uso:		**Modificar Ejercicios de Selección**	
Actores :	Programador		
Propósito:	Modificar las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad.		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la modificación de un ejercicio, especifica el ejercicio que desea realizar.			
Referencias: RF2			
Precondiciones: Se debe de tener bien definido el tipo de ejercicio que se quiere modificar.			
Poscondiciones: El ejercicio rectificado.			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere modificar.	1.1	El sistema muestra campos en los que debe introducir la pregunta con la respuesta para su modificación.
2	El programador realiza los cambios rectificando los errores tanto en la pregunta como en las respuestas dando valores nuevos según el error y oprime el botón modificar.	2.1	El sistema muestra una vista previa del ejercicio modificado.

Caso de uso:		**Modificar Ejercicios de Tabular**	
Actores :	Programador		
Propósito:	Modificar las autoevaluaciones de cada libro que se gestiona en el proyecto de Libros Electrónicos de nuestra facultad.		
Resumen: El Caso Uso se inicia cuando el programador a la hora de la modificación de un ejercicio, especifica el ejercicio que desea realizar.			
Referencias: RF2			
Precondiciones: Se debe de tener bien definido el tipo de ejercicio que se quiere modificar.			
Poscondiciones: El ejercicio rectificado.			
Curso normal de los eventos:			
Acción del actor:		Respuesta del proceso del Sistema:	
1	El programador selecciona el tipo de ejercicio que quiere modificar.	1.1	El sistema muestra campos en los que debe introducir la pregunta en la columna A y las posibles respuestas en la Columna B ,teniendo la posibilidad de en lazar la columna A con la respuesta correcta de la columna B
2	El programador realiza los cambios rectificando los errores tanto en la pregunta como en las respuestas dando valores nuevos según el error y oprime el botón modificar.	2.1	El sistema muestra una vista previa del ejercicio modificado.

2.6 Conclusiones

A partir del análisis obtenido de los requerimientos funcionales y no funcionales definidos las principales opciones del sistema, cada una con elevado nivel de especificación se determinó que la aplicación a implementar sería la forma más óptima de darle solución al problema. El sistema propuesto será una aplicación que estará compuesto por un módulo, donde se gestione la creación y modificación de las autoevaluaciones, con el objetivo tener mayor control sobre la información que vamos a brindar además de ser más factible a la hora de realizar un cambio

Capítulo 3. ANÁLISIS Y DISEÑO DEL SISTEMA.

3.1 Introducción.

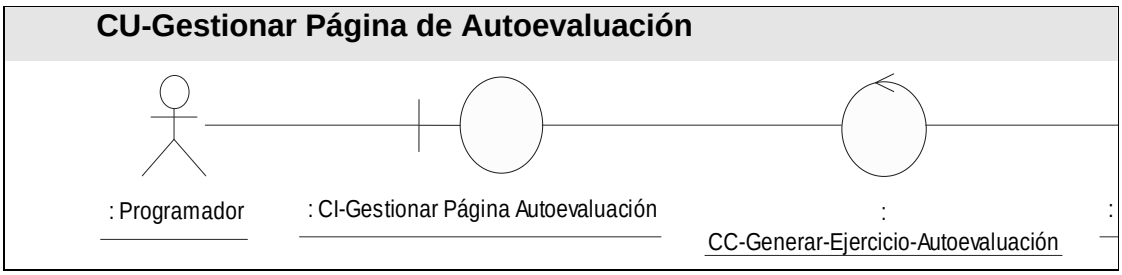
En el presente capítulo se representarán los modelos y diagramas que contribuyen a la comprensión y posterior realización de la herramienta. Representando los elementos del sistema en modelos de clases del análisis.
Se muestran los diferentes artefactos que permiten presentar la construcción de la aplicación, a través de clases usando una extensión del Lenguaje Unificado de Modelado UML. Conjuntamente se exponen los elementos que lo conforman. Posteriormente se describen los principios tenidos en cuenta en el diseño de la aplicación, tratando aspectos como su interfaz, el tratamiento de errores y la ayuda.

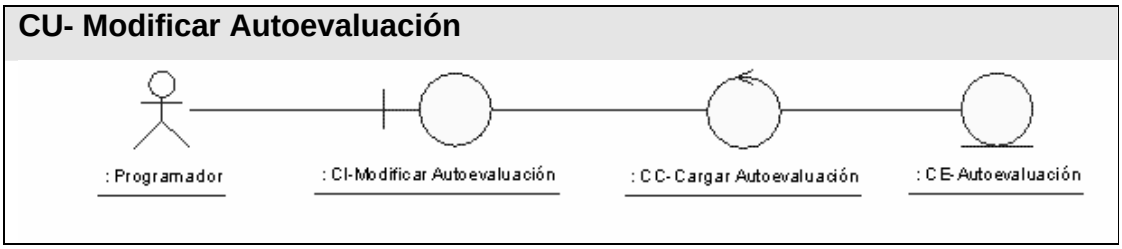
3.2. Análisis.

El análisis de la arquitectura que conformará el sistema, constituye un punto de partida para el posterior diseño, permitiendo la planificación desde etapas tempranas del proceso de desarrollo. Aunque estos diagramas representan una visión general del sistema y en algunos casos pueden obviarse, es recomendable desarrollarlos como apoyo temprano al diseño, permitiendo determinar aspectos que podrían constituir inconvenientes en etapas posteriores.

3.2.1 Diagramas de Clases del Análisis.

En este epígrafe solo se pusieron los diagramas de los casos de uso mas esenciales, los demás se encuentran en el Anexo #1.





3.3 Diseño

Durante el diseño se toman decisiones estratégicas y tácticas para cumplir los requerimientos funcionales y no funcionales de un sistema. Uno de los objetivos del diseño es crear una entrada apropiada y un punto de partida para la implementación del sistema

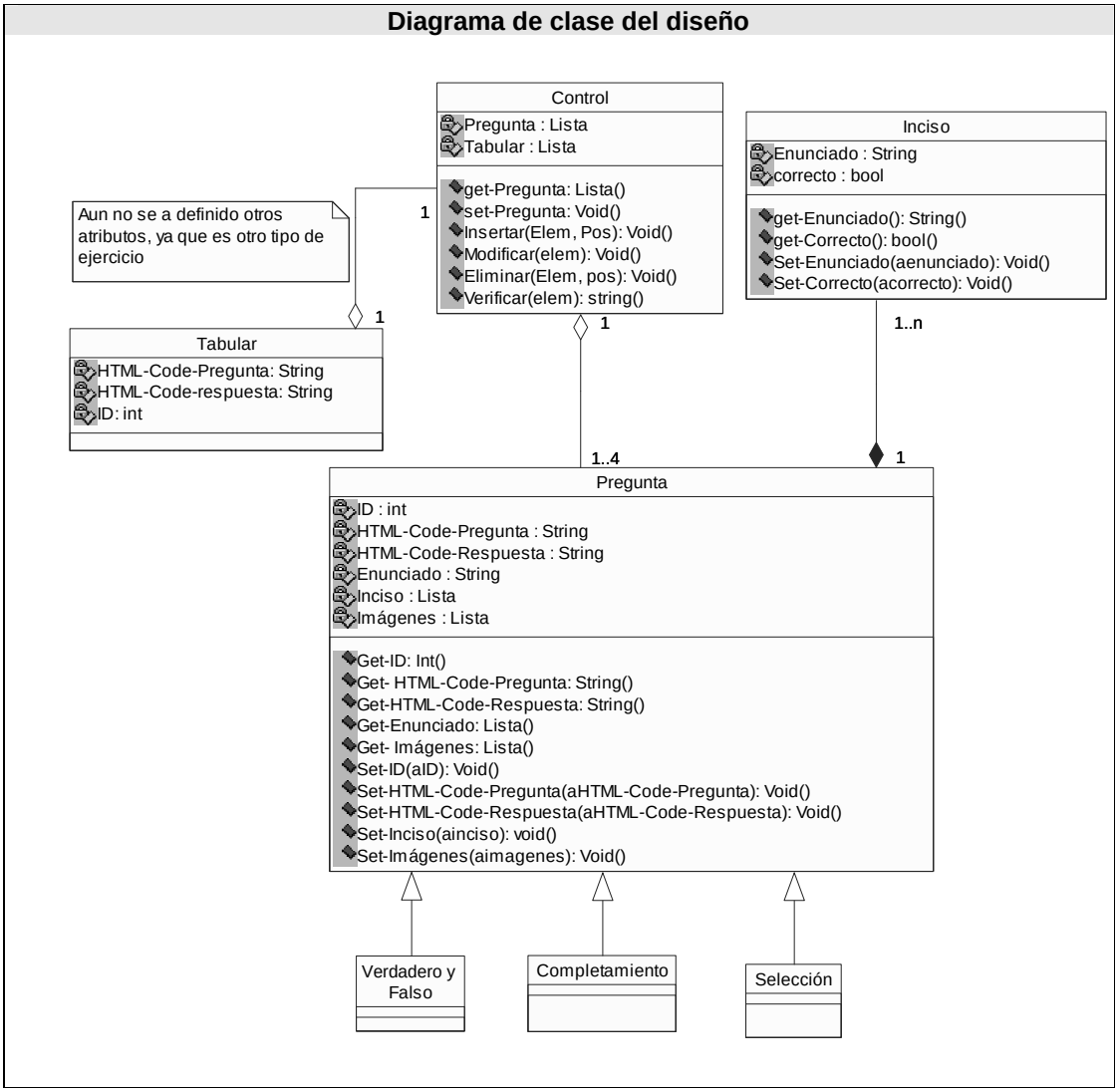
3.3.1 Modelo de clases del diseño

Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia.
En el

Anexo 2 se encuentran los diagramas de clases del diseño separados por casos de usos.

3.3.2 Diagrama de clase de diseño

Diagrama de clase del diseño



3.3.3 Descripción de las clases del diseño

Nombre: Control	
Tipo de clase controladora	
Atributo	**Tipo**
Pregunta Tabular	Lista() Lista()
Para cada responsabilidad:	
Nombre:	1-Insertar(elem, pos) 2-Modificar(elem) 3-Eliminar(elem, pos) 4-Verificar(elem) 5-Get() 6-Set(a)
Descripción:	1-Inserta un elemento dado en la posición dada. 2-Se modifica el elemento seleccionado. 3-Se elimina el elemento que se indique en su posición 4-Se verifica si el elemento buscado se encuentra en caso de que se encuentre de envía un mensaje avisando su existencia 5-Permite acceder a los datos de la clase 6-Permite cambiar o actualizar los datos de la clase

Nombre: Pregunta	
Tipo de clase entidad	
Atributo	**Tipo**
ID HTML-Code-Pregunta HTML-Code-respuesta Enunciado Inciso Imágenes	int. String String String Lista() Lista()
Para cada responsabilidad:	
Nombre:	1-Get() 2-Set(a)
Descripción:	1-Permite acceder a los datos de la clase 2-Permite cambiar o actualizar los datos de la clase

Nombre: Inciso	
Tipo de clase entidad	
Atributo	**Tipo**
Enunciado Correcto	String bool.
Para cada responsabilidad:	
Nombre:	1-Get() 2-Set(a)
Descripción:	1-Permite acceder a los datos de la clase 2-Permite cambiar o actualizar los datos de la clase

Nombre: Completamiento	
Tipo de clase entidad	
Atributo	**Tipo**
Como esta clase en hija de la clase Pregunta esta hereda todos sus atributos y funcionalidades	
Para cada responsabilidad:	
Nombre:	
Descripción:	

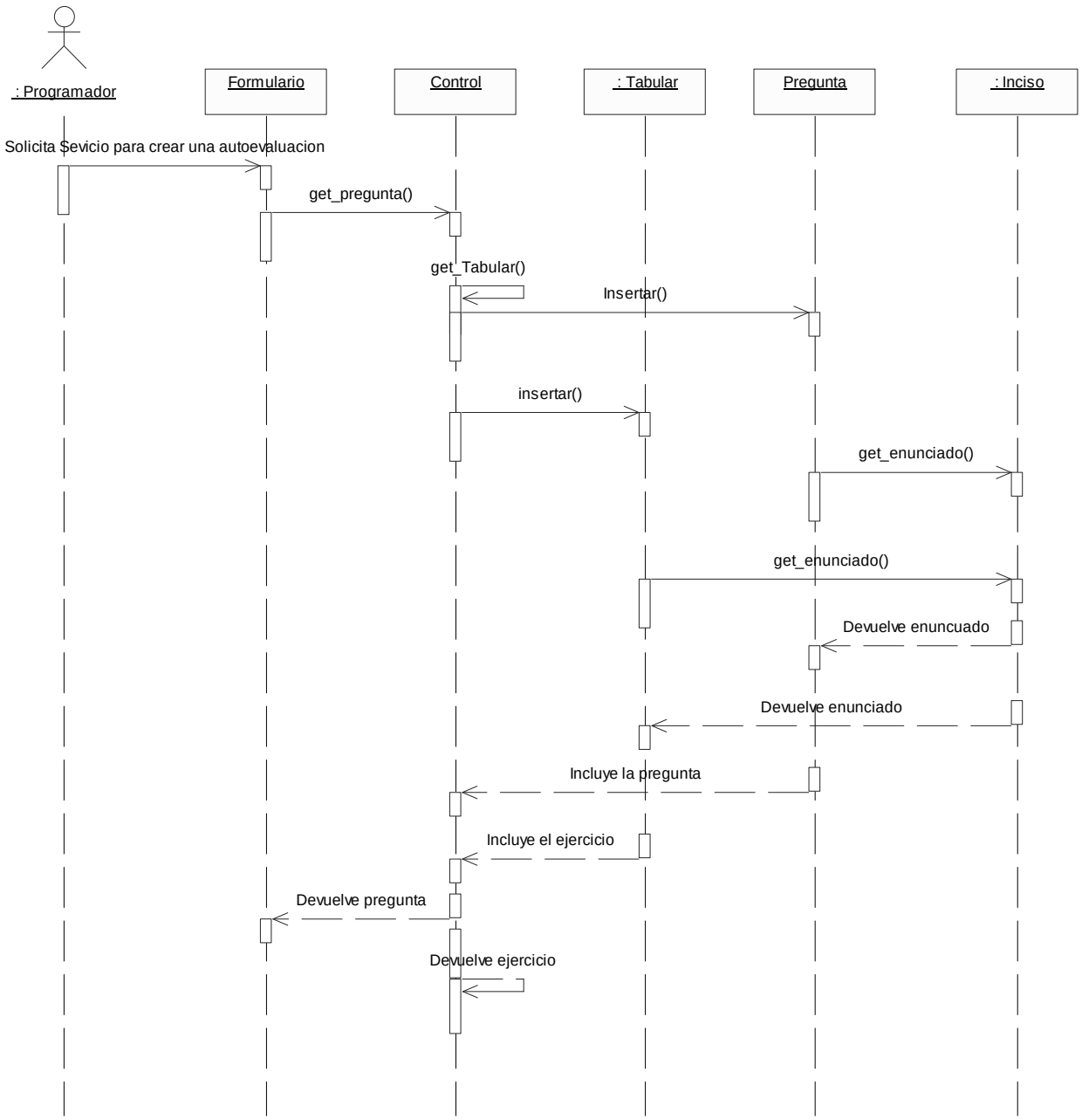
Nombre: Tabular	
Tipo de clase entidad	
Atributo	**Tipo**
ID HTML-Code-Pregunta HTML-Code-respuesta	int. String String
Para cada responsabilidad:	
Nombre:	1-Get() 2-Set(a)
Descripción:	1-Permite acceder a los datos de la clase 2-Permite cambiar o actualizar los datos de la clase

Nombre: Selección	
Tipo de clase entidad	
Atributo	**Tipo**
Como esta clase en hija de la clase Pregunta esta hereda todos sus atributos y funcionalidades	
Para cada responsabilidad:	
Nombre:	
Descripción:	

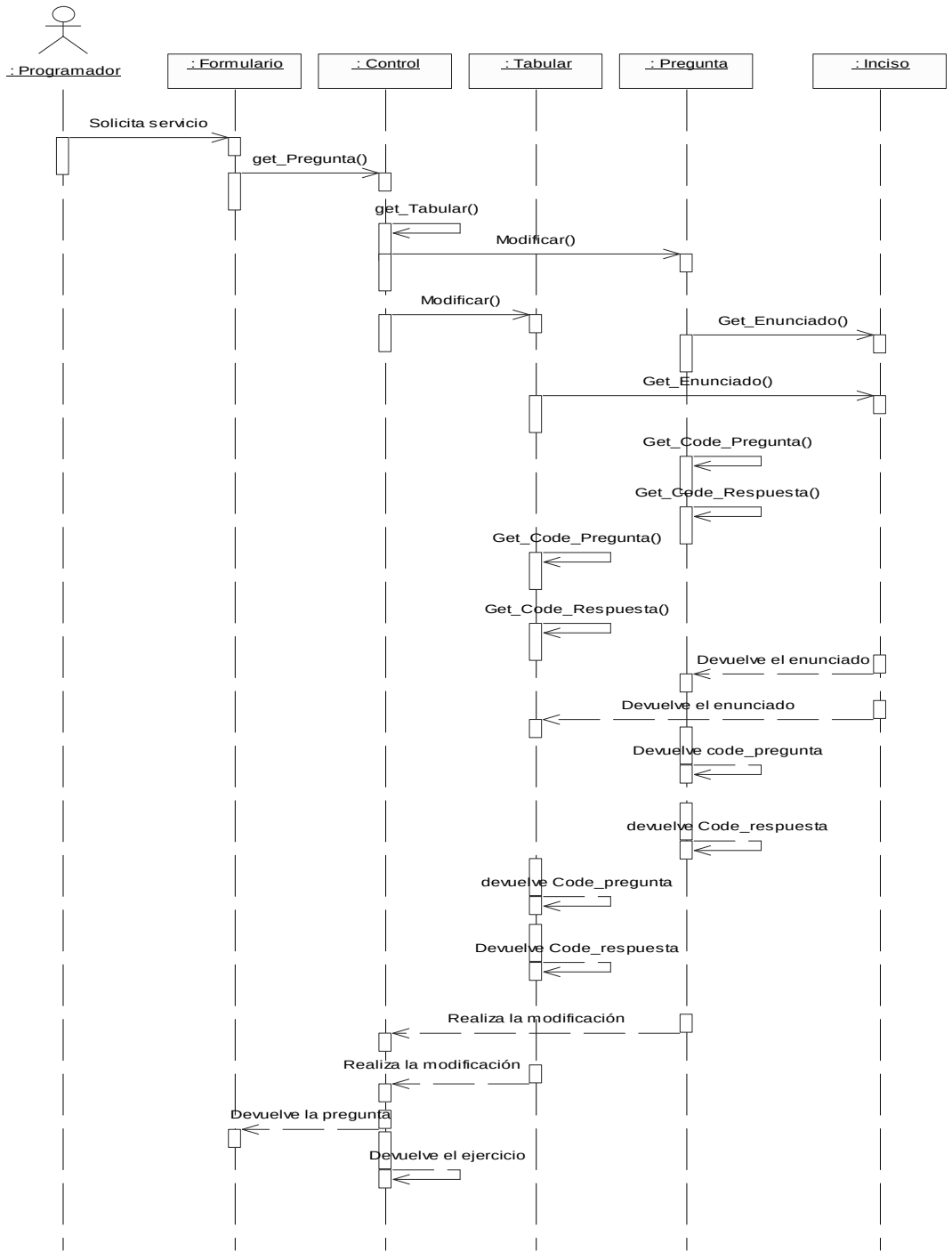
Nombre: Verdadero y Falso	
Tipo de clase entidad	
Atributo	**Tipo**
Como esta clase en hija de la clase Pregunta esta hereda todos sus atributos y funcionalidades	
Para cada responsabilidad:	
Nombre:	
Descripción:	

3.3.3 Diagramas de Interacción del Diseño

CU-Incluir Página de Autoevaluación



CU-Modificar Autoevaluación



3.3.4 Arquitectura

La Arquitectura de Software es la organización fundamental de un sistema enmarcada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución. Es a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferencia del detalle inherente a la mayor parte de las abstracciones.

3.3.4.1 ¿Qué son los patrones?

Los patrones son soluciones listas para usar, aplicables a problemas que se repiten con frecuencia en contextos acotados. Ayudan a construir la experiencia colectiva de Ingeniería de Software**,** son una abstracción de "problema – solución", ocupándose de problemas recurrentes, identificando y especificando abstracciones de niveles más altos que componentes o clases individuales y proporcionando vocabulario y entendimiento común. Algunos tipos de patrones:

- Patrones de Arquitectura. Formas de descomponer, conectar y relacionar sistemas, trata conceptos como: niveles, tuberías y filtros. Es un nivel de abstracción mayor que el de los Patrones de Diseño.
- Patrones de Programación (Idioms Patterns). Patrones de bajo nivel acerca de un lenguaje de programación concreto, describen como implementar cuestiones concretas.
- Patrones de Análisis. Conjunto de reglas que permiten modelar un sistema de forma satisfactoria.

- Patrones de Organizacionales. Describen como organizar grupos humanos, generalmente relacionados con el software.
- Otros Patrones de Software. Se puede hablar de patrones de Programación concurrente, de Interfaz Gráfica, de Organización de Código, de Optimización de Código, de Robustez de Código, de Fase de Prueba.

¿Qué es un Patrón de Arquitectura?

.Un patrón de arquitectura de software describe un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución.

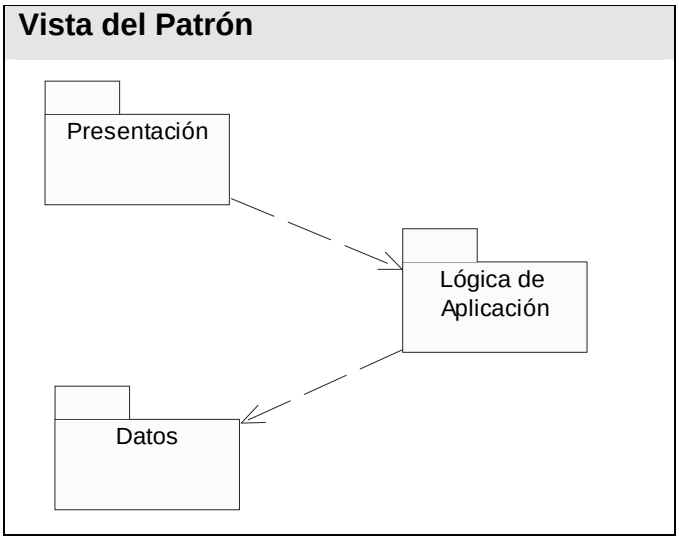
Patrón de Arquitectura 3 Capas

Capa Presentación: En la capa de presentación se establece la composición de los contenidos que se presentan finalmente al usuario, como resultado de su petición. Se añade información de presentación a los contenidos: dónde tienen que aparecer, cuál es el tamaño de letra más adecuado, dónde va el título, dónde la fotografía, dónde el gráfico, etc.

Capa Lógica de aplicación: La lógica de aplicación es la parte del portal encargada de atender las peticiones de los usuarios. Esta lógica proporciona valor añadido al contenido almacenado, combinando la información y adaptándola a las necesidades de cada usuario.

Capa Datos: En la capa de datos es donde se almacena de forma persistente toda la información necesaria para facilitar los servicios ofrecidos por el portal. El perfil de los usuarios del portal, índices de los motores de búsqueda, contenidos de agregación, información sobre la publicación, así como información sobre las terceras partes que proveen contenido o servicios al portal. La capa que se agrega es la que surge de separar definitivamente las reglas de negocio de la de Datos.

Esta arquitectura nos brinda la ventaja de aislar definitivamente la lógica de negocios de todo lo que tenga que ver con el origen de datos, ya que desde el manejo de la conexión, hasta la ejecución de una consulta, la manejará la capa de Acceso a Datos. De este modo, ante cualquier eventual cambio, solo se deberá tocar un módulo específico, así como al momento de plantear la escalabilidad de nuestro sistema, si hemos respetado las reglas básicas de diseño no deberíamos afrontar grandes modificaciones.



Ventajas y beneficios:

El uso de un modelo de n capas combinado con XML permite que los desarrollos realizados cumplan una serie de ventajas muy importantes:

- Acceso a la información en tiempo real.
- Indexación y organización de la información accesible desde una misma interfaz.
- Obtener y distribuir datos desde el mismo programa.
- Ahorro de tiempo y costes en el desarrollo de nuevas aplicaciones y la integración en el resto de los procesos de gestión de la empresa.

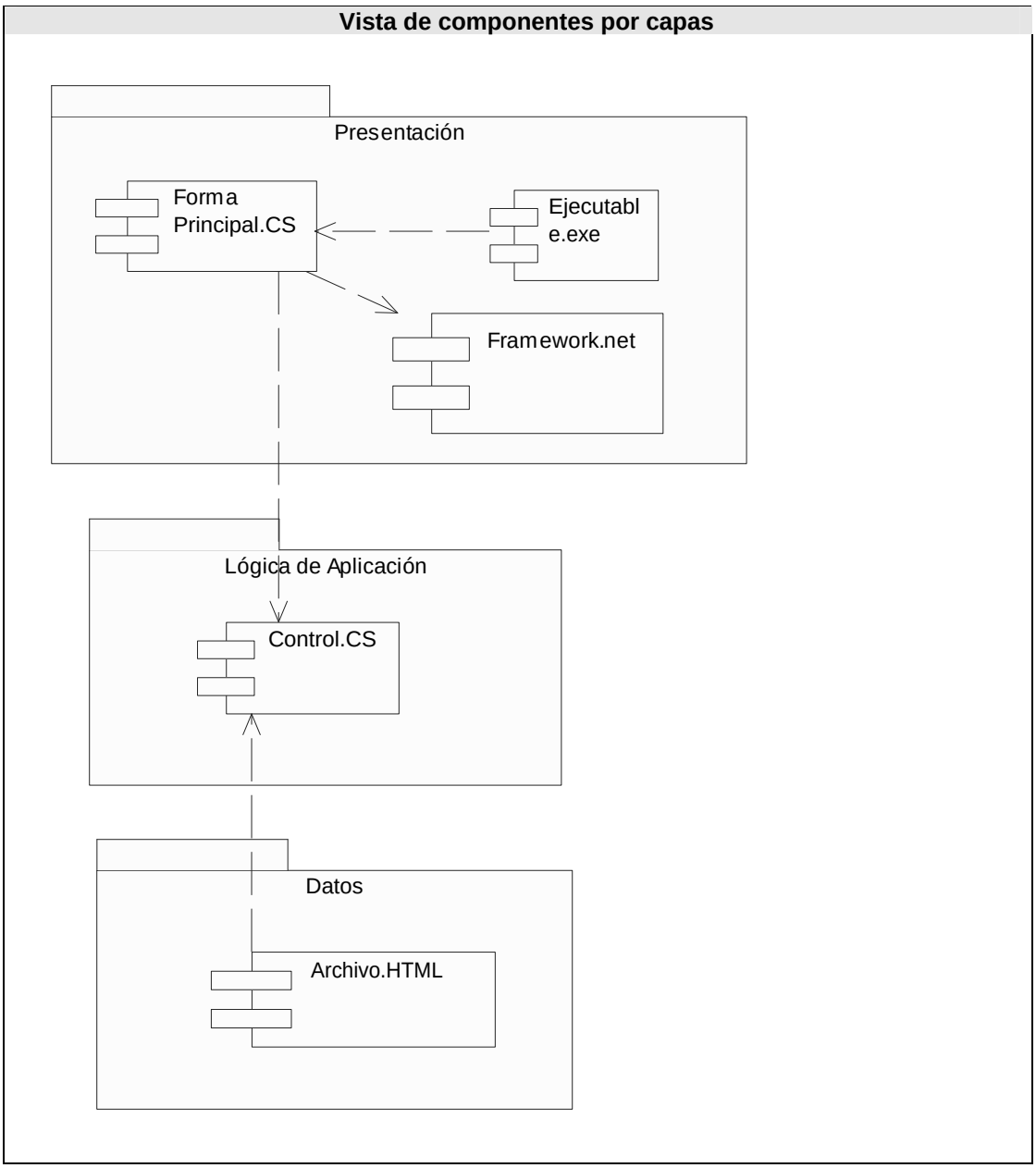
¿Qué es un Patrón de Diseño?

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios y una manera más práctica de describir ciertos aspectos de la organización de un programa. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan Los patrones de diseño pueden incrementar o disminuir la capacidad de comprensión de un diseño o de una implementación, disminuirla al añadir accesos indirectos o aumentar la cantidad de código, disminuirla al regular la modularidad, separar mejor los conceptos y simplificar la descripción. Por otro lado, los patrones de diseño, facilitan el aprendizaje al programador inexperto, pudiendo establecer parejas problema-solución. En la programación orientada a objetos resulta complicado descomponer el sistema en objetos (encapsulación, granularidad, dependencias, flexibilidad, reusabilidad, etc.), los patrones de diseño nos permitirán identificar a los objetos apropiados de una manera mucho más sencilla. También nos permitirán determinar la granularidad de los objetos.

3.3.5 Diagrama de Componentes

Se utilizan para mostrar las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre elementos de implementación y los correspondientes elementos de diseños que son implementados.

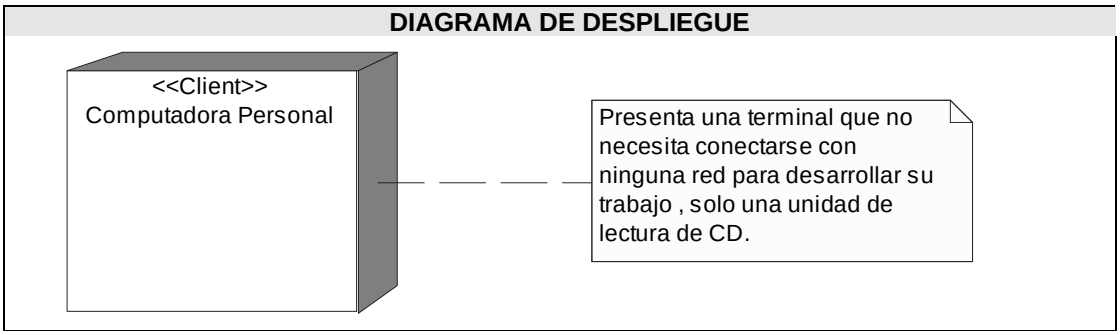
Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfases que estos soporten.



3.3.6 Diagrama de Despliegue

Muestra las relaciones físicas entre los componentes hardware *y* software en el sistema final, es decir, la configuración de los elementos de procesamiento en

tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos).



3.4 Conclusiones

En el presente capítulo se ha desarrollado el flujo de diseño a través de los Diagramas de Presentación, de Clases, así como los Diagramas de Secuencia para la realización de los Casos de Uso obtenidos en el capítulo anterior. Durante el presente capitulo se efectuó la modelación de los diagramas de clases de análisis y de diseño analizándose el dominio del problema para construir un modelo del mundo real utilizando objetos, refinando así los modelos de análisis para crear especificaciones adicionales que enriquecen al mismo con detalles próximos para una futura implementación.

Conclusiones

Después de realizar una investigación profunda sobre proyectos que se encargan de la creación de autoevaluaciones, y llevar a cabo el análisis y diseño de dicho sistema que las permita generar. Por lo que ha quedado reflejado en este documento como se lograron todos los objetivos propuestos donde se desarrollará con más precisión la implementación de dicha herramienta, logrando más facilidad a la hora de su uso en el proyecto de los Libros Electrónicos, además puede ser usado por personal ajeno al proyecto para ser más confortable su trabajo con la generación de las autoevaluaciones.

Recomendaciones

El estudio realizado en cuanto al análisis y el diseño para la creación de la herramienta se considera bastante complejo y dinámico, pero siempre se les puede hacer unas mejoras.

- Extender la investigación hasta la implementación de la herramienta y así ayudar a desarrollar los demás proyectos.

- Mejorar el diseño más sugerente logrando una interfaz atractiva.

- El uso de software libre para el desarrollo de la herramienta.

- Se recomienda que se le de importancia a la idea de que el desarrollo de la herramienta puede servir además como material de estudio y de apoyo a todos aquellas personas interesadas en la realización de autoevaluaciones. Esta puede ser utilizada donde se necesite.

- Se recomienda que un experto en el tema de la creación de las autoevaluaciones revise los contenidos obtenidos durante la investigación de esta hasta el desarrollo de la herramienta.

Referencias Bibliográficas

1. **Tamayo, I.J.L.** ***Informe de Autoevaluación 2006*****. 2007 [cited; Available from: http://www.centrogeo.org.mx/Transparencia/pdfs/informe/Informe%20AutoEvaluacion%202006.pdf.**

2. **Carretti, L.** ***welcome to English-Net*****. 2003 [cited; Available from: http://www.english-net.com.ar/area2.php?arid=28.**

3. **Lemke, C.** ***La Preparación de los Estudiantes para el Futuro*****. 2001 [cited; Available from: http://www.eduteka.org/EstMundoModerno.php.**

4. **E. Gómez Ramírez, M.P.M.M.F.E.** ***El libro electrónico: una alternativa para la mejora de la calidad en la Educación Superior*****. 2002 [cited; Available from: http://www.congreso-info.cu/UserFiles/File/Info/Info2002/Ponencias/61.pdf.**

5. **Vázquez, F.D.S.** ***HERRAMIENTAS: HOT POTATOES*****. [cited; Available from: http://209.85.207.104/search?q=cache:P0r34lGS9gMJ:www.aehermes.org/phormiga/profes/cursotic/doc/integra_IC_3h.pdf+Herramienta+HotPotatoes&hl=es&ct=clnk&cd=1&gl=cu&lr=lang_es.**

6. ***Libro electrónico*****. 2003 [cited; Available from: http://ciberhabitat.gob.mx/biblioteca/le/.**

7. **Marcos Guglielmetti, A.L., Guillem Alsina, David Yanover.** ***Herramienta*****. [cited; Available from: http://www.mastermagazine.info/termino/5234.php.**

10. **Exes.** ***Curso: Java*****. 2004 [cited; Available from: http://www.mailxmail.com/curso/informatica/java/.**

11. ***Rational Rose Enterprise Edition*****. [cited; Available from: http://www.rational.com.ar/herramientas/roseenterprise.html.**

12. **http://teleformacion.uci.cu, *Fase de inicio.Modelo de negocio*. 2007.**

8. ***Java201.com*** **2005 [cited; Available from: http://translate.google.com.cu/translate?hl=es&sl=en&u=http://www.java201.com/resources/browse/2005/javascript.html&sa=X&oi=translate&resnum=3&ct=result&prev=/search%3Fq%3DJavascript,%2B2005%26hl%3Des%26sa%3DG.**

9. ***Resumen de las características de C#*****. [cited; Available from: http://msdn.microsoft.com/es-es/library/aa287483(VS.71).aspx.**

Bibliografía Consultada

1. **Booch, G.: Rumbaugh, J. y Jacobson,** I.; *"El Lenguaje Unificado de Modelado"*. 2000. Addison-Wesley.

2. **Humphrey, Watt S.** *Introducción al Proceso Software Personal.* La Habana : Félix Varela, 2005.

3. **IVAR JACOBSON, G. B., JAMES RUMBAUGH**. *El Lenguaje Unificado de Modelado. Guía de Usuario*. 2000. p.
---. *El proceso unificado de desarrollo de software.* . 1999a. 257 p.
---. *El proceso unificado de modelado de software.*, 1999b. 208 p.

4. **Jacobson, I.; Booch, G. y Rumbaugh, J**.; *"El Proceso Unificado de Desarrollo de software"*. 2000. **Addison-Wesley**. Prólogo, Capítulos 1-5, Apéndice A. Visión General de UML, Apéndice B. Páginas 3-104, 407-424.

5. **Larman, Craig.** *UML y Patrones.* La Habana : Félix Varela, 2004.

6. **Pressman, Roger**; *Ingeniería de software*. Un enfoque práctico. 2002. McGraw-Hill/Interamericana de España. Prólogo y Capítulos 1, 2, 3, y 11. Páginas 3-48 y 181-195.

7. **Roger S. Pressman.Parte 1.** *Ingenieria del Software.* La Habana : Félix Varela, 2005.

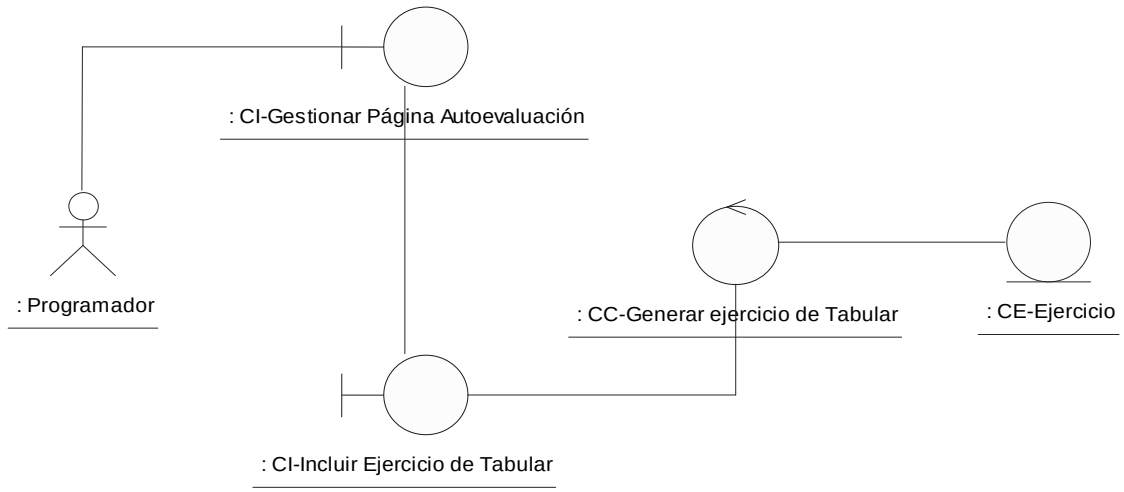
8. **Roger S.Pressman. Parte 2.** *Ingenieria del Software.* La Habana : Félix Varela, 2005.

9. **Schmuller, Joseph**. *Aprendiendo UML en 24 horas*. 1999. http://docencia.uci.cu/is/cgi-bin/list_doc_join.pl?option=77

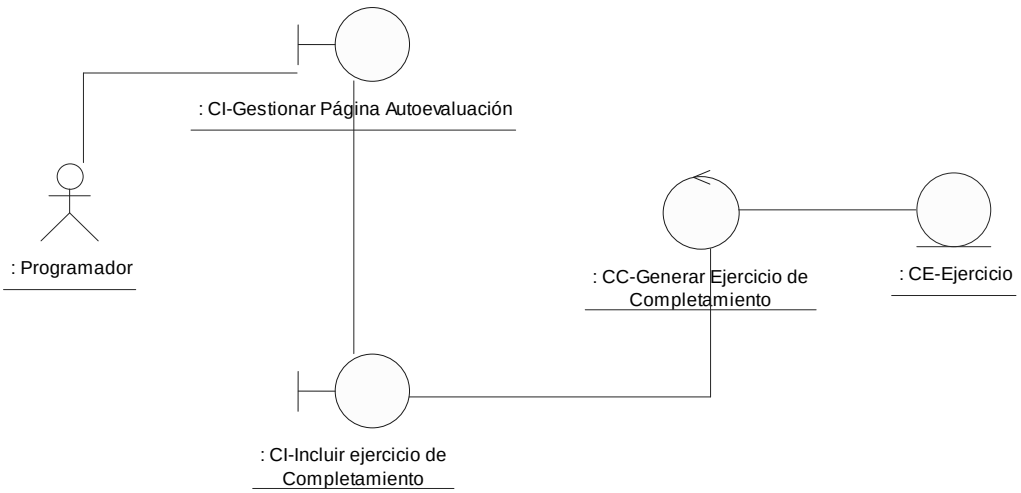
Anexos

Anexo 1. Diagramas de Clases del Análisis

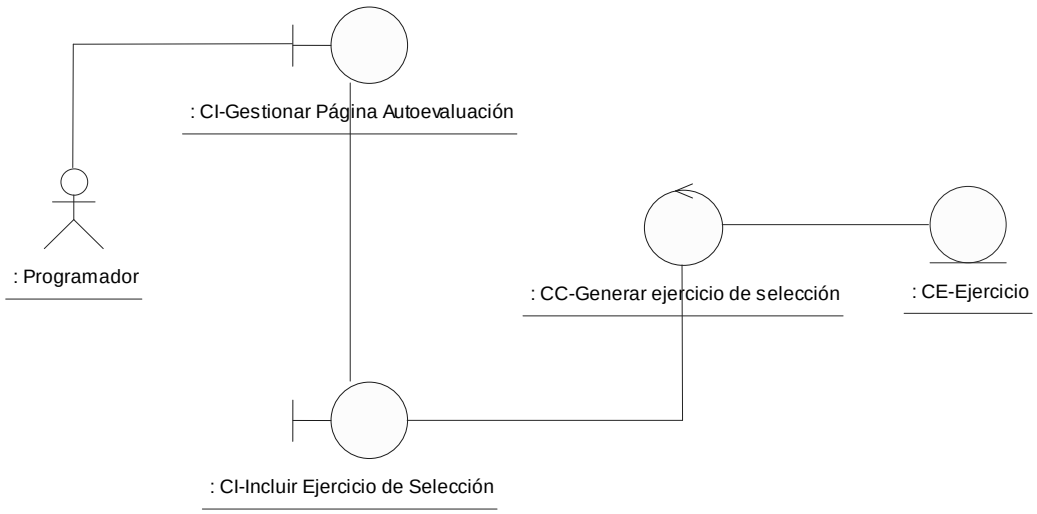
CU-Incluir Ejercicio de Tabular



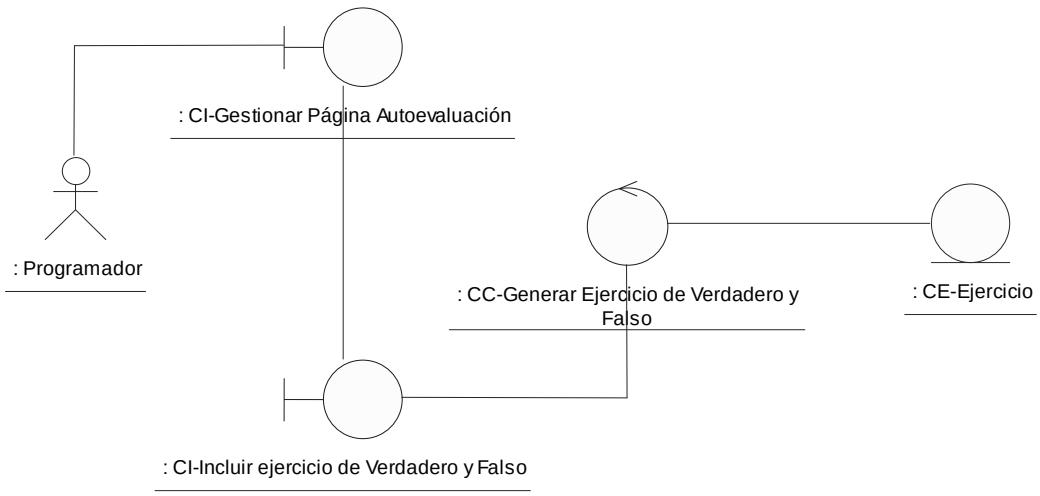
CU- Incluir Ejercicio de Completamiento



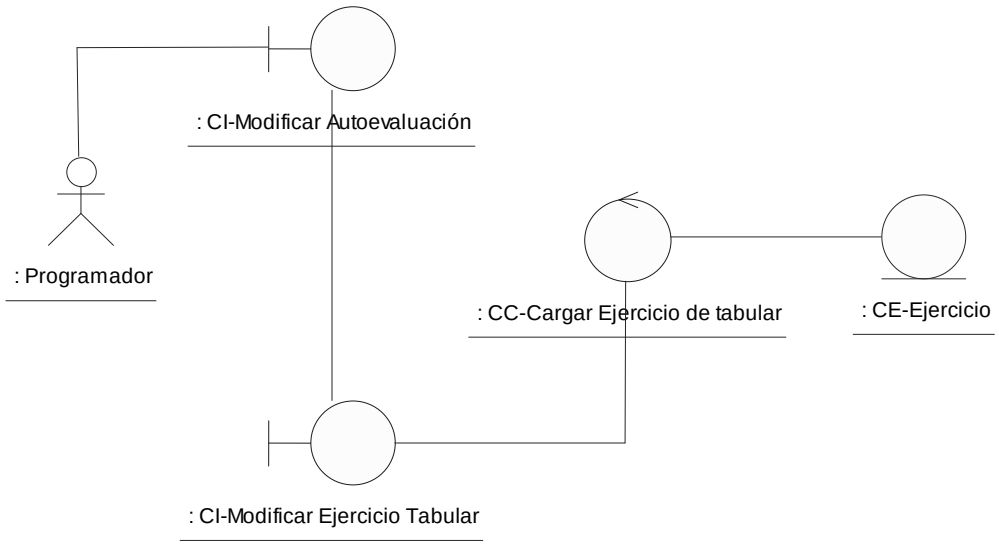
CU-Incluir ejercicio de Selección



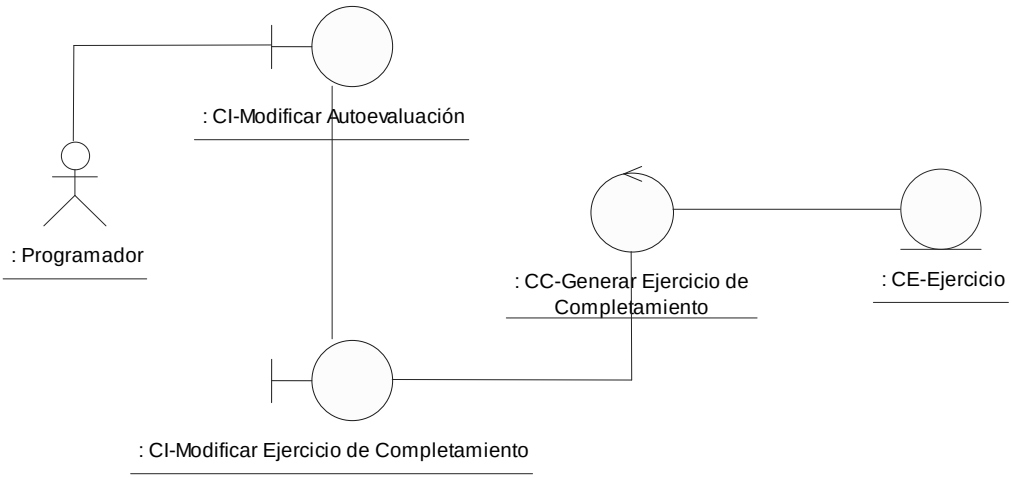
CU-Incluir Ejercicio de Verdadero y Falso



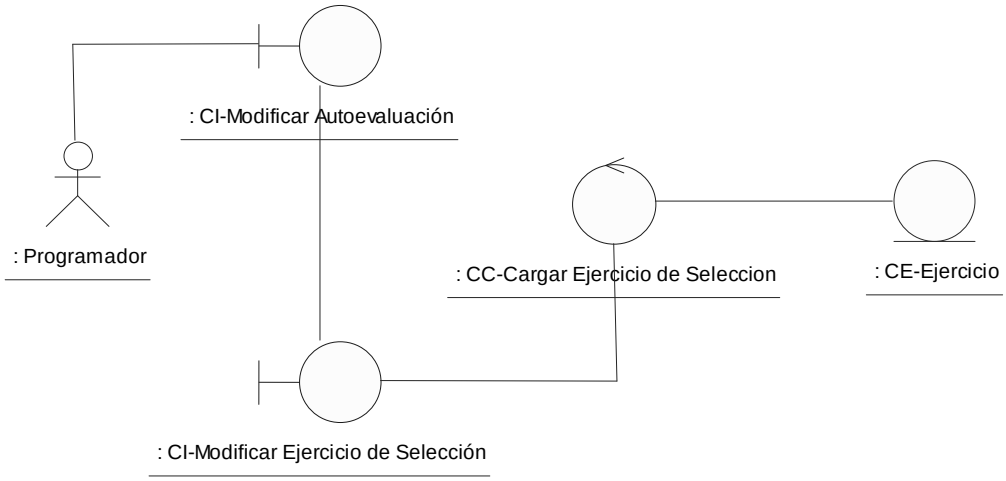
CU-Modificar Ejercicio de Tabular



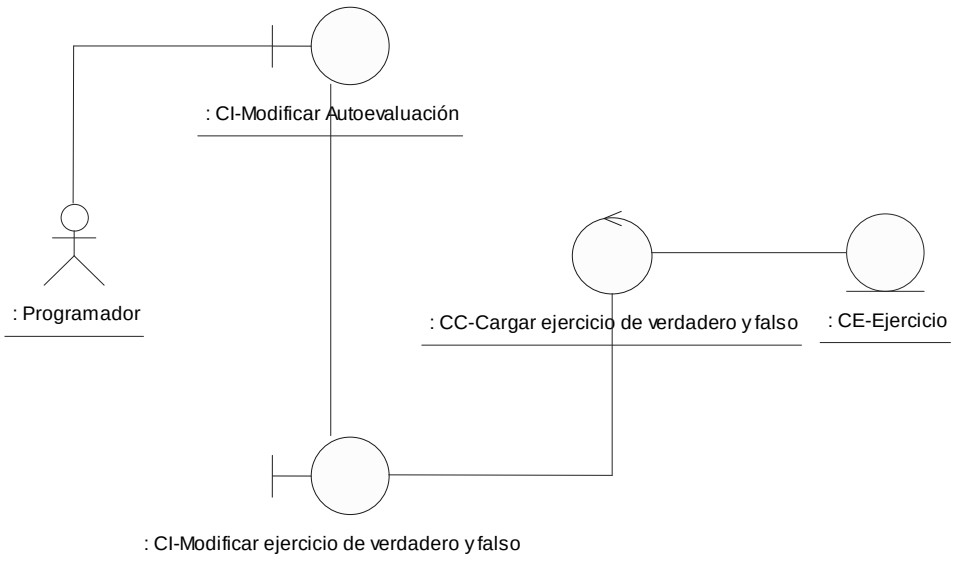
CU-Modificar Ejercicio de completamiento



CU-Modificar Ejercicio de Selección

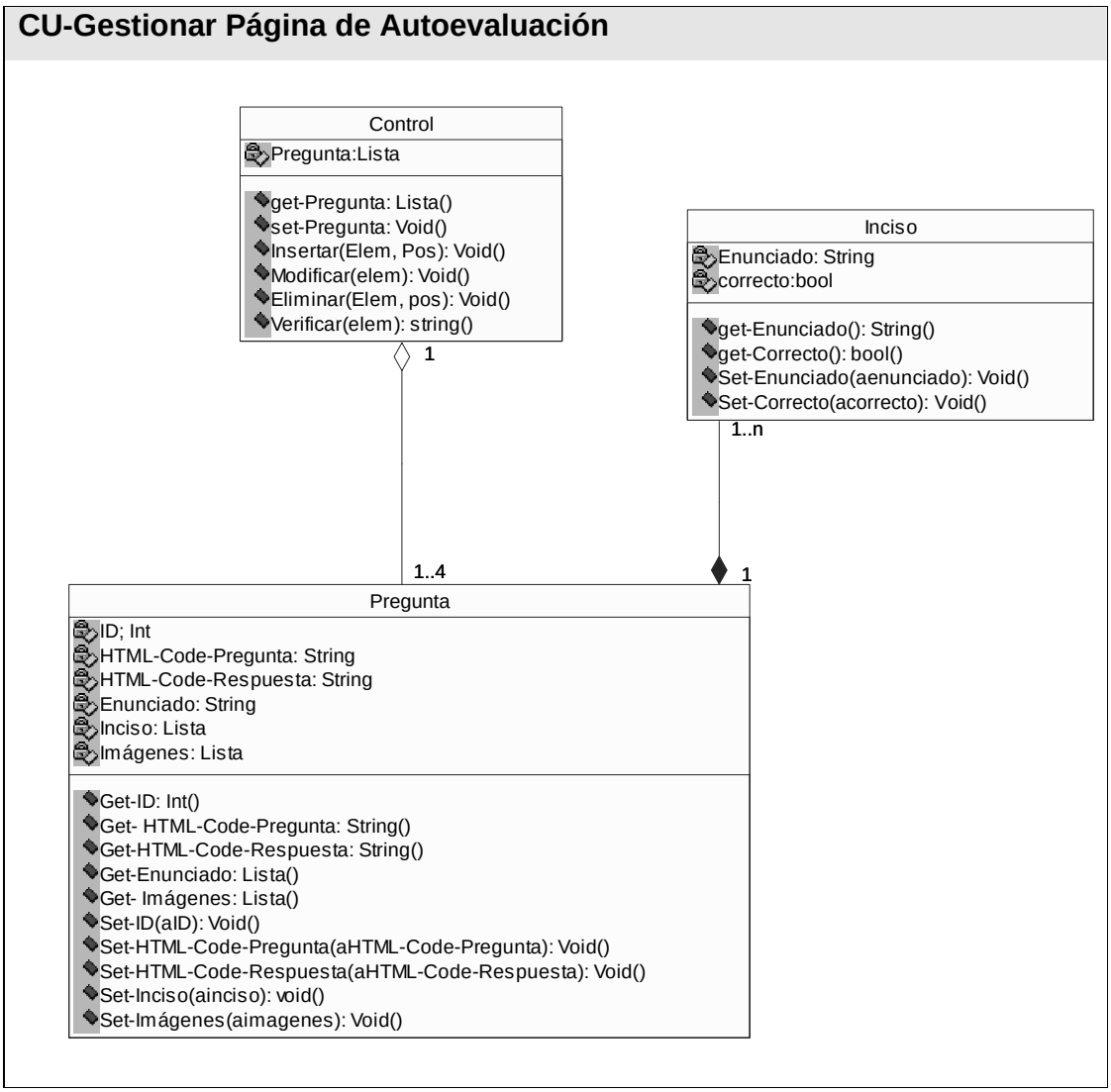


CU-Modificar Ejercicio de Verdadero y Falso

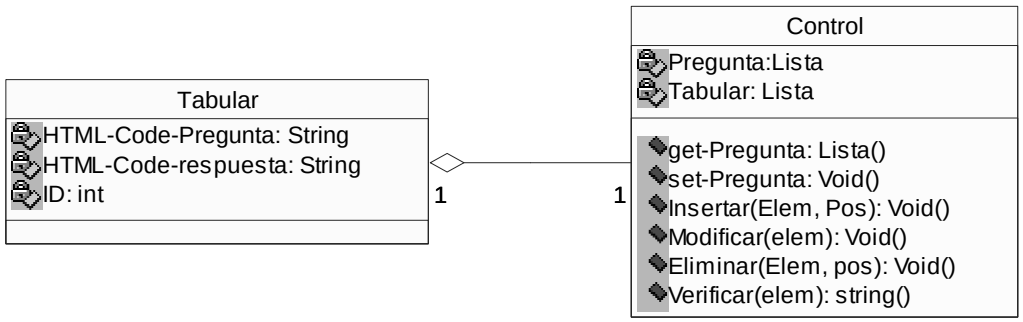


Anexo 2. Diagramas de Clases del diseño

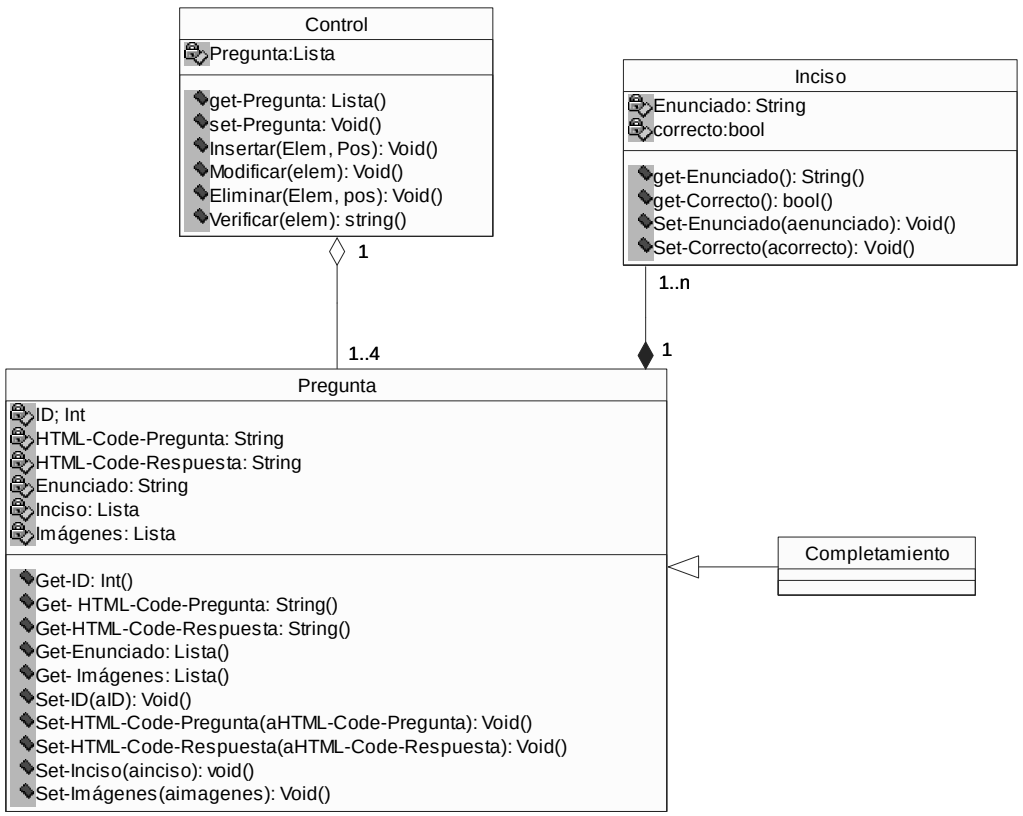
CU-Gestionar Página de Autoevaluación



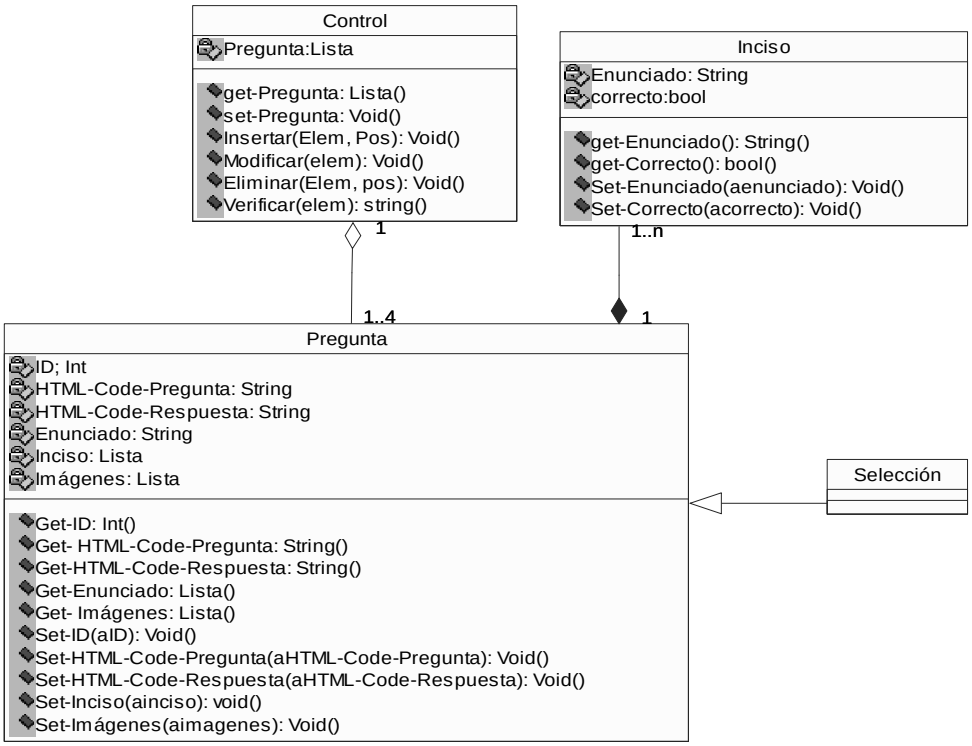
CU-Incluir Ejercicio de Tabular



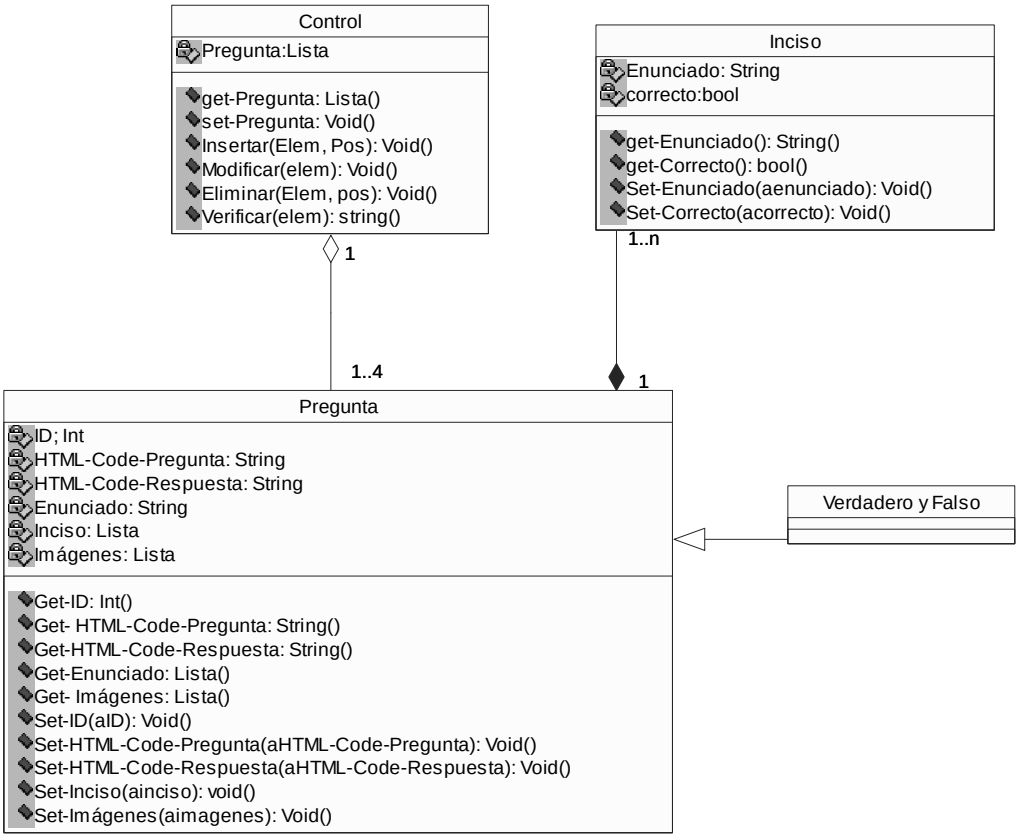
CU-Incluir Ejercicio de Completamiento



CU-Incluir Ejercicio de Selección



CU-Incluir Ejercicio de Verdadero y Falso



Glosario de términos

.

Framework: Estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado.

HTML: Lenguaje de Marcas Hipertextuales

Interfaz: Parte del programa informático que permite el flujo de información entre el programa y el usuario u otro sistema.

PHP: Acrónimo recursivo que significa **P**HP **H**ipertext **P**re-processor.

RUP: Rational Unified Process.

UML: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de sistemas de software más conocido en la actualidad.

XML: (Extensible Markup Language) Lenguaje de Marcas Extensible. Metalenguaje extensible de etiquetas desarrollado por el W3C. Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

Navegador Web: Aplicación software que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores web a través e una red. Los documentos pueden estar en cualquier dispositivo que este conectado a la computadora del usuario.

Microsoft Visual Studio .Net: Desarrollado por Microsoft a partir de 2002. Soporta los nuevos lenguajes .NET: C#, Visual Basic .NET y Managed C++, además de C++. Puede utilizarse para construer aplicaciones dirigidas a Windows (utilizando Windows Forms), Web (usando ASP.NET y Servicios Web) y dispositivos portátiles (utilizando .NET Compact Framework).

.NET Framework: Denota la infraestructura sobre la cual se reúnen un conjunto de lenguajes, herramientas y servicios que simplifican el desarrollo de aplicaciones. Es la base de la plataforma .NET.

Herramienta CASE: (Computer Aided Software Engineering) Ingeniería de Software Asistida por Ordenador. Diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.