Universidad de las Ciencias Informáticas
Centro de Entornos Interactivos 3D (VERTEX)
Facultad 5

Editor de actividades teóricas del laboratorio virtual de química

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Marlon Rodríguez Mendosa

Tutor: Ing. Jandy M. Gómez

"Año 58 de la Revolución"
La Habana, Cuba
Junio 2016

“El hombre nunca sabe de lo que es capaz hasta que lo intenta.”

Charles Dickens

Declaración de autoría

Declaro por este medio que yo Marlon Rodríguez Mendosa, soy el autor principal del trabajo final de tesis y que autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____
Marlon Rodríguez Mendosa

Autor

_____
Ing. Jandy M. Gómez

Tutor

Datos de contacto

Tutor: Ing. Jandy M. Gómez

Institución: Universidad de las Ciencias Informáticas (UCI).

Título: Ingeniero en Ciencias Informáticas

E-mail: jgomez@uci.cu

Dedicatoria

Este trabajo de diploma, está dedicado a mis padres y mis abuelos por estar en todo momento a mi lado y tener siempre una solución para mis problemas.

Agradecimientos

Agradezco a todas aquellas personas que me han apoyado durante estos 5 años de universidad, especialmente:

A mis padres por ayudarme siempre en lo que se me ocurriera.

A mis abuelos que siempre estuvieron pendientes de mí.

A mi tutor por disponer de sus conocimientos y tiempo en todo momento.

A ñañi que siempre me apoyo en los peores momentos.

A mis profesores a lo largo de estos 5 años especialmente a Jakeline y a Andy.

A los que más que compañeros llegaron a convertirse en amigos: Yandry, Fernando, Alejandro y Juan Gabriel.

Resumen

El Centro de Entornos Interactivos 3D (VERTEX) de la Universidad de las Ciencias Informáticas ha incursionado recientemente en la creación de un Laboratorio Virtual de Química, el cual constituye una herramienta poderosa para apoyar el proceso enseñanza-aprendizaje. Actualmente resulta muy difícil para los profesores la edición del contenido de los ficheros de preguntas teóricas y por otra parte la información se almacena en texto plano pudiendo ser vulnerado y afectar el resultado que se quiere lograr con el LVQ. A partir de la problemática anterior se plantea la siguiente interrogante: ¿Cómo editar y proteger los ficheros de preguntas teóricas del LVQ? Para solventarla se trazó como objetivo: desarrollar una herramienta informática que permita a los profesores de química manipular la edición y protección del fichero de preguntas teóricas en el LVQ. Para llevar a cabo la construcción del sistema fue seleccionado el Entorno de Desarrollo Integrado QTCreator, una herramienta potente para la creación de aplicaciones y el proceso de desarrollo fue guiado por la metodología de desarrollo de *software* Programación Extrema. Al concluir el trabajo se confeccionó una aplicación capaz de editar y proteger el contenido de los ficheros de preguntas teóricas del LVQ. Por último, se realizaron pruebas de aceptación para comprobar que las funcionalidades desarrolladas se comportan como esperaba el cliente.

Palabras Clave: actividades teóricas, encriptación de ficheros, Laboratorio Virtual de Química.

Índice

Índice de Ilustraciones

Índice de Tablas

Introducción

Con el transcurso de los años, el desarrollo de la ciencia informática ha permitido la implementación y perfeccionamiento de los métodos educativos, concretándose en gran medida en el *Software* Educativo (SE). La informática ha obtenido avances relevantes en la implementación de *software*. Los *Software* Educativos son aplicaciones o programas computacionales que sus características estructurales y funcionales sirven de apoyo al proceso de enseñar, aprender y administrar. Existe una gran variedad de dichos programas, dependiendo de los conocimientos que se deseen transmitir y el método que se utilice. Estos SE cuentan con la utilización de importantes recursos tales como: imágenes, modelos tridimensionales, ficheros de sonido y video, *scripts*. Estos recursos son vitales y constituyen un eslabón fundamental en el buen funcionamiento del producto [1]. La protección de estos componentes, contribuyen a la realización de productos fiables y auténticos, existen múltiples formas para hacerlo utilizando contraseñas seguras, copias de seguridad, cifrar información, entre otros.

Cuba, a tono con el panorama internacional, dedica a su vez gran esfuerzo y recursos al desarrollo de la industria del *software* educativo, necesario para lograr la informatización de la sociedad. El país ha dado un giro a favor de la producción de *software* educativo para su utilización en todas las enseñanzas, fomentando las transformaciones económicas y políticas del país en la actualidad. Existen varias entidades encargadas de desarrollar dichos *software*, entre las que se encuentran: La CUJAE, La Universidad Marta Abreu, La Universidad de Ciencias informáticas (UCI), entre otros. Uno de los centros de desarrollo de *software* de la UCI, es el Centro de Entornos Interactivos 3D, VERTEX perteneciente a la Facultad 5. Dicho centro se encuentra inmerso en la creación de un Laboratorio Virtual de Química (LVQ). Los laboratorios virtuales (LV) constituyen herramientas poderosas para apoyar el proceso de enseñanza-aprendizaje. Además están encaminados a enriquecer la preparación de los estudiantes, así como al ahorro potencial de materiales y equipos independientemente del nivel académico y del lugar donde realicen sus estudios.

El LVQ se encuentra dividido por actividades en las que se realizan ejercicios prácticos y teóricos. En el caso de las evaluaciones prácticas, el estudiante ensambla los equipos en dependencia de la actividad seleccionada y el sistema evalúa de acuerdo a los errores cometidos en el proceso de ensamblaje. Por otro lado, los ejercicios teóricos se relacionan con un conjunto de preguntas teóricas que son cargadas desde un fichero con formato *Extensible Markup Languages* (XML) o Lenguaje de Marcas Extensible. Este fichero almacena además de la pregunta, su tipo y la respuesta. Actualmente resulta muy difícil para los profesores la edición del contenido de este fichero para modificar las

preguntas teóricas puesto que no dominan el estándar XML, debido a esto no son capaces de hacer los cambios que necesiten para realizar correctamente la evaluación. Además la información se almacena en texto plano pudiendo ser vulnerado y afectar el resultado que se quiere lograr con el LVQ.

Analizando la problemática anterior, se propone como **problema de la investigación**: ¿Cómo garantizar las operaciones de modificación y protección sobre los ficheros de preguntas teóricas del LVQ?

Se precisa como **objeto de estudio**: el módulo de evaluación teórica del LVQ.

Para lo cual, se plantea como **objetivo**: desarrollar una herramienta informática que permita a los profesores de química las operaciones de modificación y protección sobre los ficheros de preguntas teóricas en el LVQ. Por lo que se determina como **campo de acción** la gestión de la información sobre ficheros de preguntas teóricas en el LVQ.

Para cumplir el objetivo se proponen las siguientes **tareas investigativas**:

- Elaboración del marco teórico de la investigación a partir del estado de la ciencia existente sobre el desarrollo de *software* educativo y específicamente sobre los LV en la actualidad.
- Identificación de los métodos que se utilizan actualmente en el aseguramiento de los ficheros para seleccionar cuál será empleado.
- Análisis y diseño del sistema para tener una guía durante el proceso de desarrollo.
- Implementación del sistema a partir del diseño realizado.
- Realización de pruebas al sistema desarrollado.

A continuación, se detallan los métodos científicos a utilizar:

Métodos Teóricos:

Histórico-lógico: Se evidencia en la necesaria profundización teórica acerca de los estudios relacionados con el desarrollo de *software* educativos y específicamente sobre los LV en la actualidad.

Analítico-Sintético: Se analizarán las informaciones adquiridas acerca de los conceptos asociados a editor y algoritmos criptográficos y sintetizar la información recopilada, permitiendo describir las características generales y las relaciones esenciales entre estas.

Modelación: Se utilizará para crear abstracciones con el objetivo de explicar la realidad, se utilizará para la modelación de los diversos diagramas necesarios en cada uno de los flujos de trabajo según la metodología seleccionada.

Métodos Empíricos:

Análisis documental: Con el objetivo de seleccionar la información necesaria para la construcción del marco teórico.

Entrevistas a especialistas: Con el objetivo de recibir los criterios de validación sobre la aplicabilidad y utilidad de lo realizado.

La presente investigación está estructurada en tres capítulos. A continuación, se muestra la estructura descripción de los capítulos:

Capítulo 1. Se expone la fundamentación teórica donde se realiza el estado de la ciencia en el cual se aborda el estudio de *software* educativos existentes que realicen las funciones de modificación y protección de ficheros. Se profundizan en conceptos relacionados con *software* educativos y algoritmos criptográficos. Además se incluye una descripción de la metodología de desarrollo que guía el proceso de creación del *software.*

Capítulo 2. Se presenta un análisis de las principales características del sistema, se exponen los artefactos generados propios de la metodología de desarrollo utilizada. Se describen los patrones de diseño y la arquitectura que fue utilizada.

Capítulo 3. Se abordan las fases de implementación y prueba de la metodología: Programación Extrema o *Extreme Programming* (XP), incidiendo fundamentalmente en las restantes fases de la metodología.

CAPÍTULO 1: Fundamentación teórica

1.1 Introducción del capítulo

El presente capítulo incluye los principales conceptos asociados al dominio del problema, un estudio del estado del arte de los algoritmos criptográficos más utilizados y las principales características de las metodologías de desarrollo de *software.*

1.2 *Software* educativo

El concepto de *software* educativo ha sido abordado por diferentes autores, atribuyéndole disímiles definiciones a pesar de las cuales se imponen las potencialidades y su absoluto basamento en los principios de la enseñanza para su vinculación en el proceso de enseñanza-aprendizaje. Es un programa creado con la finalidad específica de ser utilizado como medio didáctico, es decir, para facilitar el proceso de enseñanza-aprendizaje. El objetivo es que el intercambio sea más eficiente: incrementar la satisfacción, disminuir la frustración y, en definitiva, hacer más productivas las tareas que rodean a los alumnos [2].

Ventajas que aporta el trabajo con el SE

- Permite la interactividad con los alumnos, retroalimentando y evaluando lo aprendido, a través de ella se puede demostrar el problema como tal.
- Incide en el desarrollo de las habilidades a través de la ejercitación.
- Permite simular procesos complejos.
- Reduce el tiempo que se dispone para impartir gran cantidad de conocimientos facilitando un trabajo diferenciado, introduciendo al alumno en el trabajo con los medios computarizados [2].

Ventajas del uso del SE por parte del maestro

- Enriquece el campo de la Pedagogía al incorporar la tecnología de punta que revoluciona los métodos de enseñanza-aprendizaje.
- Pueden adaptar el *software* a las características y necesidades de su grupo teniendo en cuenta el diagnóstico en el proceso de enseñanza-aprendizaje, lo cual permite elevar su calidad.
- Permiten controlar las tareas docentes de forma individual o colectiva.
- Muestran la interdisciplinariedad de las asignaturas [2].

Tipos de *software* educativos

Existen diversos criterios referentes a las distintas clasificaciones del *software* educativo, unos se basan en las funciones didácticas de la actividad que simulan, otros en las teorías de aprendizaje en que sustentan, otros, según la forma de organización de la enseñanza que modelan. Algunas de estas clasificaciones son:

- Tutoriales.
- Entrenadores.
- Evaluadores.
- Libro electrónico.
- Simuladores (Laboratorios Virtuales) [2].

1.3 Laboratorio Virtual de Química

El LVQ está compuesto por actividades, cada una tiene dos ejercicios: un primer ejercicio teórico persigue consolidar los conocimientos adquiridos en el aula, y un ejercicio práctico que es en el que el estudiante realiza de manera virtual las prácticas de laboratorio asociadas a la actividad seleccionada. Como se muestra en la imagen (Ilustración 1) el diseño modular que se propone con este laboratorio permite, como una de sus ventajas, la posibilidad de tratar cada una de las prácticas de manera independiente. Además garantiza que se puedan agregar otras nuevas incrementando el valor agregado al *software*, tanto desde un punto de vista informático como pedagógico al aumentar el potencial didáctico.

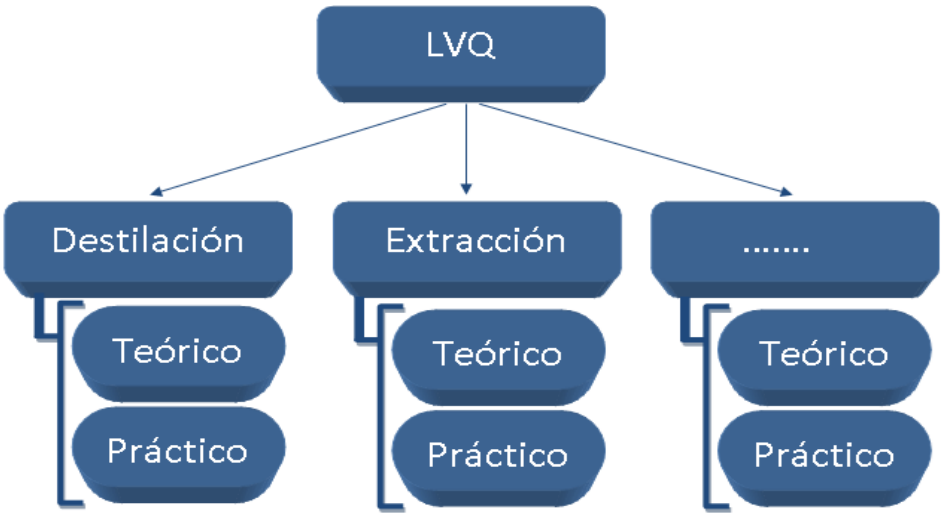

Ilustración 1. Esquema del diseño modular

1.3.1 Requerimientos informáticos del LVQ

Editor

Un editor es un programa que permite abrir y modificar archivos [3]. En instancias de la informática, el término resulta muy común como consecuencia del llamado editor de textos, que es aquel programa que permitirá redactar, editar, modificar e imprimir documentos digitales [4].

También existen los editores gráficos, estos se ocupan de la edición apoyada en equipos procesadores de imágenes digitales, en la mayoría de los casos fotos o documentos escaneados. Estas imágenes son modificadas para optimizarlas, manipularlas, retocarlas, con el fin de alcanzar la meta deseada [5].

Teniendo en cuenta las definiciones anteriores se puede definir como editor, un programa informático que permite abrir un fichero y hacer modificaciones en este, guardarlos e imprimirlos, si es necesario.

Estándar XML

XML o *Extensible Markup Language* (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Es una simplificación y una adaptación del SGML (*Standard Generalized Markup Language*) y permite definir la gramática de lenguajes específicos (de esta misma manera *HyperText Markup Language* (HTML), traducido como Lenguaje de Formato de Documentos para Hipertexto, es a su vez un lenguaje definido por SGML). XML no fue concebido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Es una tecnología sencilla que posee a su alrededor otras que hacen su complemento y a su vez la hacen mucho más grande. En la actualidad tiene gran importancia, ya que permite la compatibilidad entre sistemas para compartir información de una manera segura, fiable y fácil [7]. Algunas ventajas del estándar son:

- Es extensible, lo cual significa que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan *bugs* y se acelera el desarrollo de la aplicación.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones. Actualmente existe un grupo de

tecnologías que hacen uso de este tipo de formato para el almacenamiento de información, específicamente en el ámbito de la Realidad Virtual. Actualmente se pueden encontrar diferentes tipos de ficheros 3D, que hacen uso del formato XML para almacenar información del estado de la geometría, textura [8].

1.3.4 Bibliotecas para el trabajo con XML

Stream XML:

Esta biblioteca permite la lectura de datos de un documento XML como una secuencia de símbolos. Esto difiere de otras bibliotecas que proporcionan controladores para recibir eventos del analizador XML mientras que el Stream XML controla el ciclo, sacando símbolos del lector cuando se necesitan. Este enfoque permite construir analizadores recursivos-descendentes, permitiendo que el código de análisis XML se divida en diferentes métodos o clases [6].

DOM XML:

DOM genera un árbol jerárquico en memoria. La información en cada elemento, es considerado un nodo dentro del árbol. Este árbol jerárquico de información en memoria permite que a través de algún "*parser*" sea manipulada la información, las ventajas serían las siguientes:

- Puede ser agregado un nodo (información) en cualquier punto del árbol.
- Puede ser eliminada información de un nodo en cualquier punto del árbol.

Lo anterior se ejecuta sin incurrir en las penalidades o limitaciones de manipular un archivo de alguna otra manera [6].

1.4 Criptografía

La criptografía es el arte o ciencia de cifrar y descifrar información mediante técnicas especiales y es empleada frecuentemente para permitir un intercambio de mensajes que sólo puedan ser leídos por personas a las que van dirigidos y que poseen los medios para descifrarlos. Es una rama de las matemáticas que, al orientarse al mundo de los mensajes digitales, proporciona las herramientas idóneas para solucionar los problemas relacionados con la autenticidad y la confiabilidad. [9].

1.4.1 Algoritmos criptográficos

Todos los algoritmos actualmente basan su seguridad en la utilización de llaves, hay dos clases de algoritmos de encriptación basados en llaves, Algoritmo de clave secreta o criptografía simétrica y Algoritmo de clave pública o criptografía asimétrica. La diferencia es que los algoritmos de criptografía

simétrica utilizan la misma clave para encriptar y desencriptar, mientras que los algoritmos de criptografía asimétrica utilizan una llave distinta para la encriptación y para la desencriptación [10] [11].

DES (*Data Encryption Standard*):

Desarrollado originalmente por IBM a solicitud del NBS (del inglés *Nacional Bureau of Standard*, Oficina Nacional de Estandarización, en la actualidad llamado NIST, del inglés *National Institute of Standards and Technology*, Instituto Nacional de Estandarización y Tecnología) de Estados Unidos y posteriormente modificado y adoptado por el gobierno de ese mismo país en 1977 como estándar de cifrado de todas las informaciones sensibles no clasificadas. Este estándar de cifrado posee discrepancias en cuanto a su uso, por algunos criterios referentes a la corta longitud de su clave, y sospechas de alguna puerta trasera o *backdoor*. Se trata de un sistema de cifrado simétrico por bloques de 64 bits, de los que 8 bits (un byte) se utilizan como control de paridad (para la verificación de la integridad de la clave). Cada uno de los bits de la clave de paridad (1 cada 8 bits) se utiliza para controlar uno de los bytes de la clave por paridad impar, es decir, que cada uno de los bits de paridad se ajusta para que tenga un número impar de "1" dentro del byte al que pertenece. Por lo tanto, la clave tiene una longitud "útil" de 56 bits, es decir, realmente sólo se utilizan 56 bits en el algoritmo. En la actualidad no se ha podido romper el sistema DES criptoanalíticamente (deducir la clave simétrica a partir de la información interceptada). Sin embargo una empresa española sin fines de lucro llamado *Electronic Frontier Foundation* (EFF) construyó en Enero de 1999 una máquina capaz de probar las 2^{56} claves posibles en DES y romperlo sólo en tres días con fuerza bruta. A pesar de su caída DES sigue siendo utilizado por su amplia extensión de las implementaciones vía hardware existentes (en cajeros automáticos y señales de video, por ejemplo) [12].

VIGENÉRE:

El método original fue descrito por Giovan Batista Belaso en su libro "*La cifra del Sig. Giovan Batista Belaso"* añadió una clave repetida para cambiar cada carácter entre los diferentes alfabetos. El cifrado Vigenére es un criptosistema simétrico, es decir, utiliza la misma clave para cifrar y descifrar. Es un método de cifrado polialfabético y la llave es una secuencia de símbolos del alfabeto [13].

RIJNDAEL (AES):

El algoritmo Rijndael es un sistema simétrico de cifrado por bloques, por tanto utiliza la misma clave para el proceso de cifrado como para el proceso de descifrado. Su diseño permite la utilización de

claves de sistema con longitud variable siempre que sea múltiplo de 4 bytes. La longitud de las claves utilizadas por defecto son 128 (AES-128), 192 (AES-192) y 256 (AES-256) bits. De la misma manera el algoritmo permite la utilización de bloques de información con un tamaño variable siempre que sea múltiplo de 4 bytes, siendo el tamaño mínimo recomendado de 128 bits (el tamaño mínimo de 16 bytes) [14].

Después del análisis de los algoritmos criptográficos propuestos anteriormente, se decide utilizar el algoritmo Vigenére. Dado que puede ser implementado directamente en el código, sin tener que utilizar ninguna biblioteca. Esto permite evitar problemas de compatibilidad entre el editor de actividades teóricas y el LVQ, ya que uno esta implementado en c++ y el otro en c#. Por otra parte teniendo en cuenta que los ficheros que se desean encriptar no necesitan un nivel elevado de seguridad y por tanto el algoritmo que se utilice no debe consumir demasiados recursos de la máquina.

1.5 Metodologías de desarrollo de *software*

Las metodologías de desarrollo de *software* son un enfoque estructurado para el desarrollo de *software* que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos, son las encargadas de elaborar estrategias centradas en los clientes y equipos de desarrollo, orientadas hacia la funcionalidad y la entrega [15]. Dada la diversidad de proyectos existentes, que se pueden diferenciar de acuerdo a sus características, es coherente y preciso que existan disimiles formas de darle solución a través de las metodologías de desarrollo, las que se pueden clasificar en tradicionales y ágiles.

Las metodologías tradicionales están enfocadas en el control de proceso donde se establecen rigurosamente las actividades que se realizarán y los artefactos que se obtendrán. Estas metodologías han demostrado ser efectivas y necesarias en proyectos de larga duración, pero han presentado problemas en proyectos donde el entorno es muy cambiante y en el cual se exige reducir el tiempo de desarrollo pero sin que afecte la calidad del producto.

Las metodologías ágiles están enfocadas en el factor humano donde se establece la colaboración con el cliente y el desarrollo incremental del *software* con iteraciones muy cortas. Se caracterizan por ser flexibles al cambio y por poder reducir el tiempo de desarrollo del *software* demostrando así que es más importante contar con un *software* funcional que una documentación excesiva. Estas metodologías han demostrado tener éxito en proyectos pequeños [16].

A partir de lo analizado anteriormente, teniendo en cuenta que el equipo de desarrollo es pequeño y el sistema a desarrollar no posee un gran número de funcionalidades a implementar, se decide enfocarse en el estudio de las metodologías ágiles por ser las idóneas para este tipo de proyecto.
A continuación, se detallan las características esenciales de las metodologías ágiles más destacadas en la actualidad.

1.5.1 Metodologías de desarrollo de *software* ágiles

Programación Extrema (XP por sus siglas en inglés)

XP es una metodología ligera de desarrollo de *software* que se basa en la simplicidad, la comunicación y retroalimentación o reutilización del código desarrollado. Esta metodología se caracteriza por centrarse en fortalecer las relaciones interpersonales para el éxito del desarrollo de *software*, promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen ambiente de trabajo. XP se basa en la interacción continua entre el cliente y el equipo de desarrollo caracterizándose por mantener una conversación fluida, además de poseer soluciones implementadas simples y la capacidad de afrontar los cambios. Debido a estas particularidades XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico [17].
Algunas de las características fundamentales de XP son:

- Pruebas unitarias: se basa en las pruebas realizadas a los principales procesos con el objetivo de detectar futuros 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: 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 [18].

A continuación, se detallan las principales ventajas de la metodología XP:

- Apropiada para entornos volátiles.
- Preparada para el cambio, lo que significa reducir su coste.
- Planificación más transparente para los clientes, conocen las fechas de entrega de las funcionalidades.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- La presión está a lo largo de todo el proyecto y no en una entrega final [16].

Entre las desventajas de la metodología XP se pueden considerar las siguientes:

- No se tiene la definición de costo y el tiempo de desarrollo.
- El sistema va creciendo después de cada entrega al cliente y no se puede saber con certeza si el cliente no necesitará una funcionalidad más.
- Se requiere de la presencia constante del usuario, lo cual en la realidad es muy difícil de lograr [16].

Proceso Unificado Ágil (AUP)

El Proceso Unificado Ágil (AUP) es una versión simplificada del Proceso Unificado de *Rational* (RUP). Describe de manera fácil la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles y conceptos válidos para RUP. AUP se enfoca principalmente en la gestión de riesgos, haciendo la propuesta de que aquellos elementos con alto riesgo tengan prioridad en el proceso de desarrollo y sean tratados en etapas tempranas del mismo. El proceso establece un modelo más simple que el de RUP porque integra en una sola las disciplinas de modelado del negocio, requisitos y análisis y diseño, en el caso del resto de las disciplinas coinciden con las restantes de RUP [19].

Las principales características de la metodología AUP son:

- Abarca siete flujos de trabajos, cuatro de ingeniería y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente.
- El modelado agrupa los tres primeros flujos de RUP (Modelado del negocio, Requerimientos y Análisis y Diseño).
- Dispone de cuatro fases igual que RUP: Creación, Elaboración, Construcción y Transición.

Scrum

La metodología *Scrum* es un modelo de referencia que define un conjunto de prácticas y roles que pueden tomarse como punto de partida para el proceso de desarrollo de *software*.

Esta metodología está indicada para proyectos con un alto grado de cambios en los requisitos. Su principal característica es que el desarrollo de *software* se realiza mediante iteraciones denominadas *sprint*, donde cada *sprint* representa un incremento del producto y las reuniones diarias que se llevan a cabo a lo largo del proyecto para la coordinación e integración de las actividades a desarrollar.

Aunque *Scrum* se enfoca en la gestión de procesos de desarrollo de *software*, puede ser utilizado en equipos de mantenimiento de *software*, o en una aproximación de gestión de programas: *Scrum* de *Scrums* [20].

Las principales características de la metodología *Scrum* son las siguientes:

- Permite la creación de equipos autos organizados impulsando la co-localización de todos los miembros del equipo, y la comunicación verbal entre todos los miembros y disciplinas involucrados en el proyecto.
- Posibilita que los clientes puedan cambiar de idea sobre lo que quieren y necesitan durante el desarrollo del proyecto.
- Maximiza la capacidad del equipo de realizar las entregas rápidamente y de respuesta a los requisitos emergentes [21].

1.5.2 Fundamentación de la metodología a utilizar

Después del análisis de las metodologías propuestas anteriormente se decide utilizar la metodología XP para la realización del trabajo porque es la que mejor se adapta a las condiciones de tiempo, características cambiantes del sistema y equipo de trabajo con el que se cuenta para el desarrollo de la solución que se propone dar al problema planteado en la presente investigación. Cabe destacar que XP también es una metodología que busca la satisfacción del cliente; además se enfoca al desarrollo del producto y no a la administración del proyecto.

1.6 Selección de la herramienta CASE

Las herramientas CASE (Ingeniería de *Software* Asistida por Computadora) son aplicaciones informáticas dedicadas al aumento de la productividad en el desarrollo de *software*. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del *software* en tareas como el proceso de realizar un diseño del proyecto, cálculo de costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras [22].

Visual Paradigm

Visual Paradigm es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño, construcción, pruebas y despliegue. El *software* de modelado UML (Lenguaje de Modelado Unificado) ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite modelar todos los tipos de diagramas de clases así como generar código inverso, código desde diagramas y documentación. Es fácil de instalar y actualizar además de ser compatible entre ediciones.

Dentro de las principales características de la herramienta se encuentran: disponibilidad de múltiples versiones, de acuerdo a su necesidad, así como la capacidad de integrarse a los principales IDE

(Entorno de Desarrollo Integrado), ofrece un diseño centrado en casos de uso y enfocado al negocio de manera que genera un *software* de mayor calidad. Otra de sus características es que permite realizar ingeniería tanto directa como inversa. Además la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto. Es capaz de generar la documentación del proyecto en diferentes formatos como web o pdf y permite el control de versiones [23].

1.7 Selección del Entorno de Desarrollo Integrado

Qt Creator es un IDE multiplataforma creado por Trolltech para desarrollar aplicaciones en C++. Está basado en la biblioteca Qt y cuenta con las siguientes características principales:

- Editor avanzado para C++.
- Diseñador de formularios (GUI) integrado.
- Herramientas para la administración y construcción de proyectos.
- Completado automático.
- Depurador visual.

Actualmente Qt constituye una amplia plataforma de desarrollo que incluye un conjunto de clases, bibliotecas y herramientas para la producción de interfaces gráficas, lo que se combina perfectamente con la herramienta que se pretende desarrollar. QT hace uso del lenguaje C++ como lenguaje de programación y a su vez es capaz de operar en varias plataformas por lo que resulta ideal al cumplir con las políticas de desarrollo de *software* libre de la UCI. Esta biblioteca está provista de una amplia gama de herramientas que facilitan la creación de formas, botones y ventanas de diálogo con el uso del mouse, lo que permite la creación de aplicaciones muy elegantes, lo que hace más amigable la interacción de usuarios con las aplicaciones desarrolladas [24].

Está provista de tres grandes ventajas ante otras bibliotecas similares y que la rivalizan a su vez:

- Qt es completamente gratuito para aplicaciones de código abierto.
- Es una biblioteca que está disponible para casi todas las plataformas, ya sea Sistemas Unix, Linux, MacOS, Solaris, así como para la familia Windows, por lo que una aplicación puede ser compilada y utilizada en cualquier plataforma sin necesidad de cambiar el código.
- Qt hace uso de una extensa biblioteca de clases y herramientas para la creación de aplicaciones gráficas, estas bibliotecas de clases están bien documentadas, así como de fácil uso y tienen una gran herencia de programación orientada a objetos [24].

1.8 Consideraciones parciales del capítulo

Después de haber realizado un estudio del estado del arte se concluye que el análisis de los principales conceptos asociados al dominio del problema permitió tener una mejor comprensión del sistema que se va a desarrollar. Para llevar a cabo una correcta ingeniería de *software*, se selecciona XP como metodología de desarrollo, ya que garantiza la generación de artefactos que son necesarios para una mayor comprensión de la aplicación a desarrollar. Además se escogió el algoritmo Vigenére con el objetivo de proteger el contenido de los ficheros de preguntas teóricas. De este modo se está en condiciones de desarrollar el editor de actividades teóricas del laboratorio virtual de química, utilizando las herramientas y tecnologías seleccionadas.

CAPÍTULO 2: Características y diseño del sistema.

2.1 Introducción del capítulo

En este capítulo se realiza un análisis de las principales características del sistema, se exponen los artefactos generados propios de la metodología de desarrollo utilizada para la implementación de la solución que se propone. Además, se describe la arquitectura y los patrones de diseño que fueron utilizados.

2.2 Propuesta de sistema

Como propuesta de solución técnica de este trabajo se establece: el desarrollo de una aplicación con interfaz gráfica de usuario que permita la edición y protección de los ficheros de evaluación teórica del LVQ. Para lograr lo antes mencionado la aplicación debe garantizar un grupo de funcionalidades que son imprescindibles para su correcto funcionamiento.

Principales funcionalidades

- Brindar la posibilidad de visualizar las preguntas de los ficheros de evaluación teórica.
- Permitir la modificación de las preguntas de los ficheros de evaluación.
- Asegurar la protección del contenido de los ficheros de evaluación.

Formas de salvar y leer

El fichero contendrá los datos almacenados en formato XML, para ello se hará uso de las clases para la manipulación de XML que brinda QT específicamente la *Stream XML y DOM que* por su forma de manipular los ficheros, se acopla perfectamente con las tareas que se quieren lograr.

Las reglas del negocio no son más que parámetros o restricciones por los que se rige el sistema, siendo necesario para el desarrollo de la solución informática.

- El procedimiento de lectura de los ficheros se hará solamente a archivos con extensión .XML.
- Las rutas de los ficheros deben ser definidas, por medio de la herramienta.
- Los ficheros deben ser encriptados con anterioridad.
- Los ficheros serán desencriptados por la aplicación final.

2.3 Características no funcionales del sistema

Los requerimientos no funcionales (RNF) son propiedades o cualidades que debe tener el producto. Estas propiedades son las características que hacen al producto atractivo, usable, rápido o confiable. Entre los requerimientos no funcionales del sistema propuesto se encuentran:

Usabilidad:

RNF 1: Los botones utilizados en la aplicación deben tener un nombre o descripción entendible para el usuario.

RNF 2: Debe existir una distribución y categorización de los contenidos que facilite la navegación al usuario.

RNF 3: La aplicación estará dirigida a los educadores como usuarios finales.

RNF4: La navegación debe ser sencilla tanto para usuarios con conocimientos avanzados de informática como para los más inexpertos.

Restricciones del diseño y la implementación:

RNF 5: Como entorno de desarrollo integrado se utilizará QTCreator en su versión 3.1.

RNF 6: Como lenguaje de programación se utilizará C++.

RNF 7: Se utilizará como fuente de almacenamiento ficheros en formato xml.

Hardware:

RNF 8: Es necesaria como mínimo una PC con CPU Intel Pentium IV, 512 Mb de RAM O cualquier PC con prestaciones superiores.

***Software*:**

RNF9: El sistema debe permitir su ejecución en Sistema Operativo Windows y GNU/Linux.

2.4 Fase I: Exploración

La fase de Exploración es el espacio que crea XP para que los clientes expongan las Historias de Usuario (HU) de mayor relevancia para el negocio, por lo que se convierte en la primera entrega del producto. A la vez el equipo de desarrollo se adapta a las tecnologías, herramientas y prácticas que se disponen en el proyecto, se exploran las posibilidades de la arquitectura del

sistema construyendo un prototipo. Esta fase consta de pocas semanas y está en dependencia del grado de familiaridad que tengan los programadores con la tecnología [25].

Historias de Usuario

Las HU es la técnica que usa la metodología XP para la especificación de requisitos del *software*. Estas cobran mayor protagonismo en la fase de Exploración donde los clientes plantean las HU en lenguaje natural e intentan modelar los requisitos desde la perspectiva de los usuarios. Las HU deben poseer la suficiente información como para determinar cuál se implementará primero, además de posibilitar la realización de estimaciones razonables sobre la duración de cada HU (23). El dinamismo y la flexibilidad de las que se valen las HU, son claves en el momento de enfrentar algún tipo de modificación en los requisitos. Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores, utilizando como medida el punto. Un punto, equivale a una semana ideal de programación (5 días laborables). Las historias generalmente valen de 1 a 3 puntos [26]. A continuación, se describen las historias de usuario definidas.

Tabla 1. HU 1: Cargar fichero.

Historia de Usuario	
Número: 1	**Nombre:** Cargar fichero
Usuario: Profesor de LVQ	**Iteración asignada:** 1
Prioridad en el Negocio: Alto	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: El flujo comienza cuando el profesor de LVQ decide cargar el fichero de preguntas en el editor, para ello debe seleccionar la opción Abrir, posteriormente debe elegir el fichero que desee. Acto seguido, el fichero de preguntas es desencriptado por el editor mediante el algoritmo criptográfico Vigenére.	
Observaciones: Para poder cargar un fichero, este debe cumplir con las reglas del negocio.	

Tabla 2 HU 2: Visualizar preguntas.

Historia de Usuario	
Número: 2	**Nombre:** Visualizar preguntas
Usuario: Profesor de LVQ	**Iteración asignada:** 1
Prioridad en el Negocio: Alta	**Puntos estimados:** 1
Riesgo en desarrollo: Alto	**Puntos reales:** 1
Descripción: El flujo comienza luego de que el profesor de LVQ carga el fichero, donde se le permitirá una visualización de las preguntas teóricas presentes en dicho fichero.	
Observaciones: Para poder visualizar las preguntas deberá haber cargado el fichero anteriormente.	

Tabla 3. HU 3: Administrar preguntas.

Historia de Usuario	
Número: 3	**Nombre:** Administrar preguntas
Usuario: Profesor de LVQ	**Iteración asignada:** 2
Prioridad en el Negocio: Alta	**Puntos estimados:** 2.5
Riesgo en desarrollo: Alto	**Puntos reales:** 2.5
Descripción: El flujo comienza cuando el profesor de LVQ decide realizar alguna acción con las preguntas presentes en el fichero, para lo cual tendrá la posibilidad de modificar pregunta, modificar respuesta y eliminar.	
Observaciones: Para poder modificar las preguntas deberá haber cargado el fichero anteriormente.	

Tabla 4 HU 4: Adicionar preguntas.

Historia de Usuario	
Número: 4	**Nombre:** Adicionar preguntas
Usuario: Profesor de LVQ	**Iteración asignada:** 2
Prioridad en el Negocio: Alta	**Puntos estimados:** 2
Riesgo en desarrollo: Alto	**Puntos reales:** 2
Descripción: El flujo comienza cuando el profesor de LVQ decide adicionar una nueva pregunta para ello debe seleccionar la opción adicionar pregunta, donde el editor mostrará una ventana que le permitirá dependiendo del tipo seleccionado adicionar la nueva pregunta.	
Observaciones:	

Tabla 5 HU 5: Guardar cambios.

Historia de Usuario	
Número: 5	**Nombre:** Guardar cambios
Usuario: Profesor de LVQ	**Iteración asignada:** 3
Prioridad en el Negocio: Alta	**Puntos estimados:** 1.5
Riesgo en desarrollo: Alto	**Puntos reales:** 1.5
Descripción: El flujo comienza cuando el profesor de LVQ decide guardar los cambios para ello debe seleccionar la opción guardar o guardar como. Acto seguido, el fichero de preguntas es encriptado por el editor mediante el algoritmo criptográfico Vigenére.	
Observaciones: El fichero no se guardará si no posee al menos diez preguntas.	

2.5 Fase II: Planificación

En esta fase se evidencia uno de los valores que acoge XP para el desarrollo de un sistema, como es el caso de la comunicación. Lo cual se confirma cuando el cliente establece la prioridad de cada HU, y a raíz de esto, los programadores realizan una estimación del esfuerzo necesario que conlleva desarrollar cada una de ellas. Además, se toman acuerdos sobre el contenido de la primera entrega y se acuerda la confección de un cronograma en conjunto con el cliente [27].

Estimación de esfuerzo por historia de usuario

Se realiza la estimación de esfuerzo que arroja cada historia de usuario con el objetivo de obtener un correcto desarrollo del sistema. En la tabla 6 se muestra la estimación realizada:

Tabla 6. Estimación de esfuerzo por historia de usuario

Historia de Usuario	Puntos de Estimación(semanas)
Cargar fichero	1
Visualizar preguntas	1
Administrar preguntas	2.5
Guardar los cambios	1.5
Adicionar ejercicios	2

2.6 Fase III: Iteraciones

Esta es la fase principal en el ciclo de desarrollo de XP. Las funcionalidades son desarrolladas en esta fase, generando al final de cada una un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las HU no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. El cliente, por lo tanto, también debe participar activamente durante esta fase del ciclo.

Plan de Duración de las iteraciones

A continuación, se presenta el plan de duración de las iteraciones. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las historias de usuario en cada iteración como se muestra en la tabla 7:

Tabla 7. Plan de Duración de las iteraciones

Iteración	Descripción	Historias de Usuario asignadas	Duración total en semanas
1	Se implementan las historias de usuario iniciales, dando al sistema las funcionalidades básicas necesarias y estableciendo una primera versión entregable al cliente.	Cargar fichero Visualizar preguntas	2
2	Se implementan un grupo de funcionalidades necesarias para lograr la robustez de la solución propuesta.	Administrar preguntas Adicionar ejercicios	4.5
3	Con esta iteración se completan las funcionalidades necesarias para completar la solución.	Guardar los cambios	1.5

Plan de Entregas

En el plan de entrega que se describe en la tabla 8 se hace una propuesta de las versiones del sistema. Cada versión se hará al finalizar una iteración.

Tabla 8. Plan de entrega de versiones

Entregable	iteración 1	iteración 2	iteración 3
Editor de Actividades Teóricas del LVQ.	**Versión 0.4** **30/4/2016**	**Versión 0.8** **7/6/2016**	**Versión 1.0** **19/6/2016**

2.7 Diseño del sistema

Las tarjetas CRC (Cargo o clase, Responsabilidad y Colaboración) identifican y organizan las clases bajo el paradigma orientado a objetos (lo que incluye asignación de responsabilidades), cada tarjeta contiene el nombre de la clase (que representa una o más historias de usuario), una descripción de las responsabilidades o métodos asociados con la clase, así como la lista de las clases con que se relaciona o que colaboran con ella. Las tarjetas CRC son el único trabajo de diseño que se genera como parte del proceso de XP [28].

Tabla 9. Prototipo de tarjeta CRC

Nombre de la clase	
Responsabilidades	Colaboradores

Clase: Se refiere a las clases persistentes del sistema a desarrollar.

Responsabilidades: Función que realiza la clase dentro del sistema.

Colaboradores: Relación que tiene la clase con otras clases persistentes del sistema.

A continuación se muestran las tarjetas CRC pertenecientes al editor de actividades:

Tabla 10. Tarjeta CRC - XmlManager

Tarjeta CRC	
Clase: XmlManager	
Responsabilidades: Leer configuración Salvar configuración	**Colaboradores:** ExercisesSM ExercisesSU ExercisesToF

Tabla 11. Tarjeta CRC - MainWindow

Tarjeta CRC	
Clase: MainWindow	
Responsabilidades: Abrir fichero Mostrar ejercicios Guardar	**Colaboradores:** BusinnessManager XmlManager WidgetSelectionMultiple WidgetSelectionOne WidgetTrueOrFalse AddExerciseWindow

Tabla 12. Tarjeta CRC – ExercicesToF

Tarjeta CRC	
Clase: ExercicesToF	
Responsabilidades: Contener los datos de las preguntas del tipo verdadero o falso	**Colaboradores:** Exercises

Tabla 13. Tarjeta CRC – ExercicesSU

Tarjeta CRC	
Clase: ExercicesSU	
Responsabilidades: Contener los datos de las preguntas del tipo selección única	**Colaboradores:** Exercises

Tabla 14. Tarjeta CRC – ExercicesSM

Tarjeta CRC	
Clase: ExercicesSM	
Responsabilidades: Contener los datos de las preguntas del tipo selección múltiple	**Colaboradores:** Exercises

Las restantes tarjetas confeccionadas se encuentran en los anexos del trabajo de diploma.

2.8 Arquitectura de *software*

Se entiende por arquitectura de *software* la representación de alto nivel de la estructura de un sistema o aplicación. Describe las partes que la integran, las interacciones entre ellas, los patrones que supervisan su composición y las restricciones a la hora de aplicar esos patrones [29]. En el campo de la ingeniería de *software*, el concepto de arquitectura ha sido fundamental como estrategia para enfrentar la complejidad del desarrollo de soluciones informáticas y establecer acuerdos de diseño de alto nivel para estas. En una arquitectura de *software* confluyen tres elementos fundamentales:

- Los modelos que definen la estructura, topología y dinámica del sistema.
- La trazabilidad o correspondencia de dichos modelos con los requisitos o necesidades establecidas en el contexto que va a operar la solución.
- Las reglas, los principios y justificaciones que rigen la arquitectura y que sustentan las decisiones que se tomaron.

La arquitectura de *software* es fundamental a la hora de garantizar que la solución cumpla con los criterios de calidad establecidos en los requisitos [30].

Modelo Vista Controlador

Modelo Vista Controlador (MVC) es un estilo de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. El modelo lo constituyen los datos con los que trabaja la aplicación y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- El Modelo: es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.
- La Vista: es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.
- El Controlador: es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo [31].

A continuación en la Ilustración 2 se presenta la aplicación del patrón arquitectónico Modelo-Vista-Controlador en el Editor de Actividades Teóricas del LVQ.

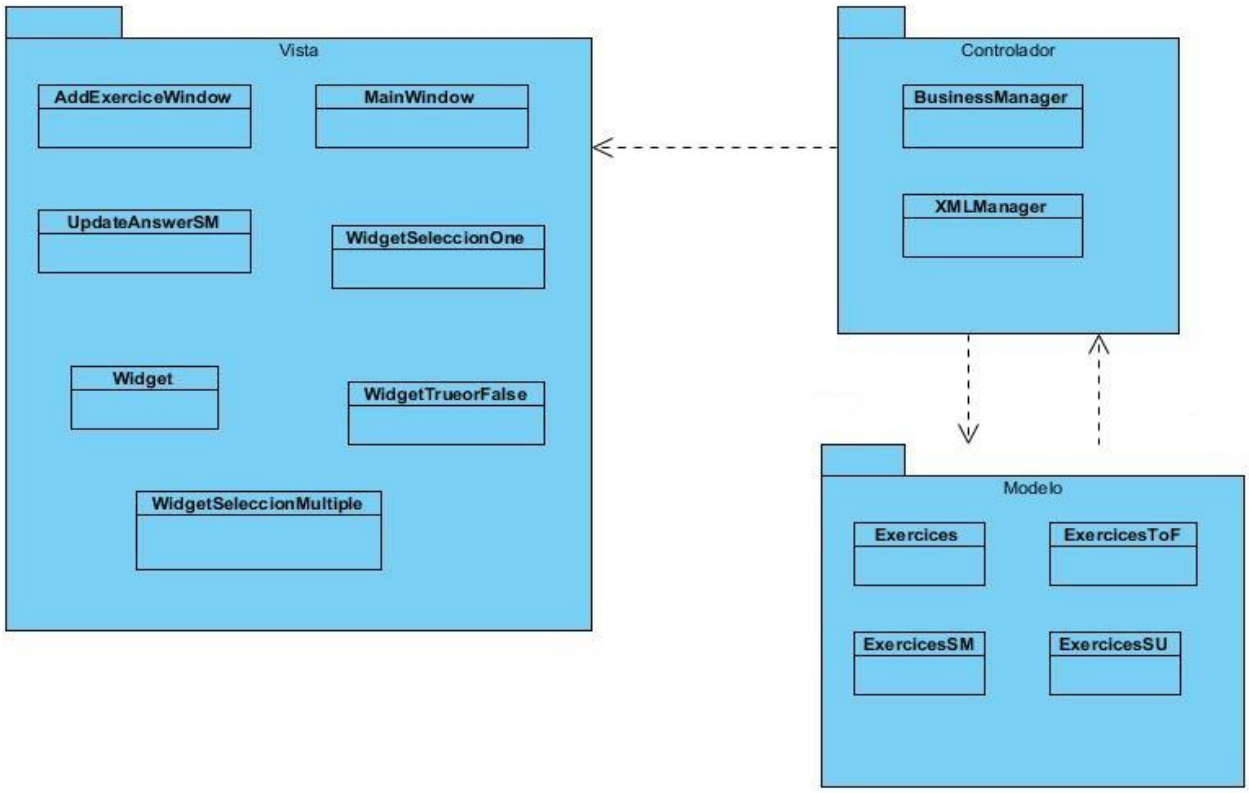

Ilustración 2. Aplicación del patrón arquitectónico Modelo-Vista-Controlador.

2.9 Patrones de diseño

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de *software* [32]. Lograr un diseño simple, pero a la vez robusto requiere del empleo de buenas prácticas. A continuación, se detallan los patrones utilizados durante el desarrollo del sistema.

Patrones GOF

Los patrones de la "pandilla de los cuatro" (por sus siglas en inglés GOF) son un conjunto de 23 patrones agrupados a partir de dos criterios: propósito y alcance. Las categorías definidas son: creacionales, estructurales y de comportamiento [33]. A continuación se detalla el patrón que se utilizó en el diseño:

Patrón Singleton

El patrón Singleton asegura que exista una única instancia de una clase. A primera vista, uno puede pensar que pueden utilizarse clases con miembros estáticos para el mismo fin. Sin embargo, los resultados no son los mismos, ya que en este caso la responsabilidad de tener una única instancia recae en el cliente de la clase. El patrón Singleton hace que la clase sea responsable de su única instancia, quitando así este problema a los clientes.

Adicionalmente, si todos los métodos de esta clase son estáticos, éstos no pueden ser extendidos, desaprovechando así las capacidades polimórficas que nos proveen los entornos orientados a objetos. El funcionamiento de este patrón es muy sencillo y podría reducirse a los siguientes conceptos:

- Ocultar el constructor de la clase Singleton, para que los clientes no puedan crear instancias.
- Declarar en la clase Singleton una variable miembro privada que contenga la referencia a la instancia única que se desea gestionar.
- Proveer en la clase Singleton una función o propiedad que brinde acceso a la única instancia gestionada por el Singleton. Los clientes acceden a la instancia a través de esta función o propiedad.

Estas reglas se cumplen en todas las implementaciones del Singleton, independientemente de los recaudos que deban tomarse para soportar la correcta ejecución en entornos multihilo. El ciclo de vida de los Singleton es un aspecto importante a tener en cuenta [33].

Utilización: Se evidencia en la clase BusinnessManager.

Patrones GRASP

Los Patrones Generales de Asignación de Responsabilidades (por sus siglas en inglés GRASP) son patrones de diseño utilizados para la asignación de responsabilidades a una determinada clase. El grupo GRASP está conformado por 5 patrones principales y 4 adicionales aplicables al diseño orientado a objetos [33]. A continuación se refleja la utilización de estos en la solución que se propone:

Experto: Consiste en la asignación de responsabilidades al más competente en información, la clase cuenta con la información necesaria para cumplir la responsabilidad. Es el principio básico de asignación de responsabilidades que suele utilizarse en el diseño Orientado a Objetos.

Utilización: se evidencia en la clase ExercicesSU ya que es la experta en realizar todas las funcionalidades referentes a ese tipo de pregunta.

Alta Cohesión: Asignar una responsabilidad de modo que la unión se mantenga a gran escala. Asignar a las clases responsabilidades que trabajen sobre una misma área de aplicación y que no tengan mucha complejidad. Mejoran la claridad y facilidad con que se entiende el diseño.

Utilización: las clases: Exercices, ExercicesToF, ExercicesSM y ExercicesSU poseen una alta cohesión puesto que son las que se encargan del manejo íntegro de los XML.

Bajo Acoplamiento: Asignar una responsabilidad para mantener un engranaje pobre. Es un principio que se debe recordar durante las decisiones de diseño. Soporta el diseño de clases más independientes. Asigna las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible

Utilización: se establecieron las relaciones únicamente necesarias entre las clases para garantizar un bajo acoplamiento.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Asigna la responsabilidad del manejo de mensajes de los eventos de un sistema a una clase.

Utilización: la clase XMLManager es la encargada de controlar toda la información de los xml.

Consideraciones parciales del capítulo

En el capítulo se detallaron las principales particularidades del funcionamiento del Editor de Actividades Teóricas. Se expusieron los artefactos generados en las fases que establece la metodología XP. Con la utilización de esta metodología aparecen factores primordiales para el éxito del producto, como el diálogo entre clientes y desarrolladores. Ya establecida una solución propuesta para el sistema, y definidos aspectos como la arquitectura y los patrones de diseño a utilizar, queda establecida la vía para llevar a cabo la implementación y prueba del sistema.

CAPÍTULO 3: Implementación y prueba del sistema

3.1 Introducción del capítulo

En este capítulo se realiza una valoración de las fases de construcción y prueba de la metodología XP. Se exponen las tareas de ingeniería generadas para cada historia de usuario que se desarrolló. Además se exponen las pruebas de aceptación efectuadas al sistema.

3.2 Implementación

En esta fase se descomponen las HU en tareas de ingeniería, estas pueden estar descritas por un lenguaje técnico y no ser necesariamente entendible por el cliente. Tienen como objetivo definir cada una de las actividades que dan cumplimiento a las HU, de forma tal que se entienda lo que el sistema tiene que hacer y facilite su construcción [25]. A continuación se describen algunas de las tareas de ingeniería correspondientes a las HU del sistema.

Tabla 15. Tarea de ingeniería 1 para HU 1

Tarea	
Número Tarea: 1	**Número Historia:** 1
Nombre tarea: Cargar fichero	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.2
Fecha inicio: 18/4/2016	**Fecha fin:** 19/4/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite cargar el fichero de evaluación teórica.	

Tabla 16. Tarea de ingeniería 2 para HU 1

Tarea	
Número Tarea: 2	**Número Historia:** 1

Nombre tarea: Desencriptar fichero	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha inicio: 20/4/2016	**Fecha fin:** 22/4/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite desencriptar el fichero de evaluación teórica.	

Tabla 17. Tarea de ingeniería 1 para HU 2

Tarea	
Número Tarea: 3	**Número Historia:** 2
Nombre tarea: Leer configuración	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.4
Fecha inicio: 25/4/2016	**Fecha fin:** 27/4/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite leer las preguntas del fichero para luego proceder a visualizarlas.	

Tabla 18. Tarea de ingeniería 2 para HU 2

Tarea	
Número Tarea: 4	**Número Historia:** 2
Nombre tarea: Visualizar preguntas	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.6
Fecha inicio: 27/4/2016	Fecha fin: 29/4/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite visualizar las preguntas del fichero teniendo en cuenta los tres tipos de preguntas existentes.	

Tabla 19. Tarea de ingeniería 1 para HU 3

Tarea	
Número Tarea: 5	**Número Historia:** 3
Nombre tarea: Modificar pregunta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha inicio: 2/5/2016	Fecha fin: 5/5/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite modificar la pregunta escogida del fichero.	

Tabla 20. Tarea de ingeniería 2 para HU 3

Tarea	
Número Tarea: 6	**Número Historia:** 3
Nombre tarea: Actualizar respuesta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.2
Fecha inicio: 6/5/2016	Fecha fin: 13/5/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite actualizar la respuesta escogida del fichero.	

Tabla 21. Tarea de ingeniería 3 para HU 3

Tarea	
Número Tarea: 7	**Número Historia:** 3
Nombre tarea: Eliminar pregunta	

Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 16/5/2016	Fecha fin: 18/5/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite eliminar la pregunta escogida del fichero.	

Tabla 22. Tarea de ingeniería 1 para HU 4

Tarea	
Número Tarea: 8	**Número Historia:** 4
Nombre tarea: Adicionar pregunta del tipo verdadero o falso	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 18/5/2016	Fecha fin: 20/5/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite adicionar una pregunta del tipo verdadero o falso.	

Tabla 23. Tarea de ingeniería 2 para HU 4

Tarea	
Número Tarea: 9	**Número Historia:** 4
Nombre tarea: Adicionar pregunta del tipo selección única	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.8
Fecha inicio: 23/5/2016	Fecha fin: 26/5/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite adicionar una pregunta del tipo selección única.	

Tabla 24. Tarea de ingeniería 3 para HU 4

Tarea	
Número Tarea: 10	**Número Historia:** 4
Nombre tarea: Adicionar pregunta del tipo selección múltiple	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1.2
Fecha inicio: 27/5/2016	Fecha fin: 7/6/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite adicionar una pregunta del tipo selección múltiple.	

Tabla 25. Tarea de ingeniería 1 para HU 5

Tarea	
Número Tarea: 11	**Número Historia:** 5
Nombre tarea: Guardar cambios	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 8/6/2016	Fecha fin: 10/6/2016
Programador responsable: Marlon Rodríguez Mendosa	
Descripción: Permite guardar los cambios realizados en el fichero.	

Tabla 26. Tarea de ingeniería 2 para HU 5

Tarea	
Número Tarea: 12	**Número Historia:** 5
Nombre tarea: Encriptar fichero	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 13/6/2016	Fecha fin: 18/6/2016
Programador responsable: Marlon Rodríguez Mendosa	

Descripción: Permite encriptar el fichero.

3.3 Pruebas

Uno de los pilares de XP es el proceso de pruebas. XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. XP divide las pruebas del sistema en dos grupos:

- Pruebas unitarias: encargadas de verificar el código y diseñadas por los programadores.
- Pruebas de aceptación o pruebas funcionales: destinadas a evaluar si al final de una iteración se consiguió la funcionalidad requerida diseñada por el cliente final.

Las pruebas de aceptación son de gran importancia dado que significan la satisfacción del cliente con el producto desarrollado y el final de una iteración y el comienzo de la siguiente por esto, el cliente es la persona adecuada para diseñar las pruebas de aceptación [35].

Pruebas de caja negra mediante pruebas de aceptación

El objetivo de estas pruebas es verificar los requisitos, por este motivo, los propios requisitos del sistema son la principal fuente de información a la hora de construir los Casos de Pruebas de Aceptación (CPA). Las pruebas de aceptación son creadas a partir de las historias de usuario. Durante una iteración la historia de usuario seleccionada en la planificación de iteraciones generará pruebas de aceptación. El cliente o usuario especifica los aspectos que serán objetos de prueba cuando una historia de usuario ha sido correctamente implementada. Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento y no se considera completa hasta que no supera estas [35].

Una prueba de aceptación es como una caja negra. Cada una de ellas representa una salida esperada del sistema. Es responsabilidad del cliente verificar la corrección y toma de decisiones acerca de estas pruebas [36]. A continuación se muestran las pruebas de aceptación a realizarse en cada iteración.

Iteración 1:

Tabla 27. CPA Cargar fichero.

Caso de prueba de aceptación
Historia de usuario: Cargar fichero
Nombre: Cargar fichero
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Para cargar el fichero se debe especificar la ruta mediante la aplicación.
Entradas/Pasos de ejecución: Seleccionar en el menú la opción, Abrir y aparece una ventana para seleccionar el fichero que se quiere abrir, se especifica la ruta y presionar el botón Aceptar.
Resultado esperado: Se carga el fichero especificado.
Evaluación de la prueba: Satisfactoria

Tabla 28. CPA Visualizar preguntas.

Caso de prueba de aceptación
Historia de usuario: Visualizar preguntas
Nombre: Visualizar preguntas
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Para visualizar las preguntas se debe cargar el fichero anteriormente.
Entradas/Pasos de ejecución: Luego de haber seleccionado el fichero a cargar, acto seguido, el editor muestra las preguntas que posee el fichero**.**
Resultado esperado: Se visualizan las preguntas.
Evaluación de la prueba: Satisfactoria

Iteración 2:

Tabla 29. CPA Administrar preguntas.

Caso de prueba de aceptación
Historia de usuario: Administrar preguntas.
Nombre: Administrar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: El usuario debe seleccionar la pregunta que quiere modificar.
Entradas/Pasos de ejecución: Seleccionar en la opción, modificar pregunta, aparecerá un formulario para llenar los campos y presionar el botón Aceptar.
Resultado esperado: Se modifica la pregunta correctamente.
Evaluación de la prueba: Satisfactoria

Tabla 30. CPA Administrar preguntas.

Caso de prueba de aceptación
Historia de usuario: Administrar preguntas.
Nombre: Administrar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: El usuario debe seleccionar la respuesta que quiere actualizar.
Entradas/Pasos de ejecución: Seleccionar la respuesta que se desea actualizar y después presionar la opción Aceptar.
Resultado esperado: Se actualiza la respuesta correctamente.

Evaluación de la prueba: Satisfactoria

Tabla 31. CPA Administrar preguntas.

Caso de prueba de aceptación
Historia de usuario: Administrar preguntas.
Nombre: Administrar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: El usuario debe seleccionar la pregunta que quiere eliminar.
Entradas/Pasos de ejecución: Seleccionar la pregunta que se desea eliminar, presionar la opción eliminar y seleccionar el botón Aceptar.
Resultado esperado: Se elimina la pregunta correctamente.
Evaluación de la prueba: Satisfactoria

Tabla 32. CPA Adicionar preguntas.

Caso de prueba de aceptación
Historia de usuario: Adicionar preguntas.
Nombre: Adicionar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución:
Entradas/Pasos de ejecución: Seleccionar la opción del menú Adicionar preguntas, se selecciona el tipo verdadero o falso, aparece un formulario con los campos correspondientes y después de completarlos seleccionar el botón Aceptar.
Resultado esperado: Se adiciona una pregunta del tipo verdadero o falso correctamente.

Evaluación de la prueba: Satisfactoria

Tabla 33. CPA Adicionar preguntas.

Caso de prueba de aceptación
Historia de usuario: Adicionar preguntas.
Nombre: Adicionar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución:
Entradas/Pasos de ejecución: Seleccionar la opción del menú Adicionar preguntas, se selecciona el tipo selección única, aparece un formulario con los campos correspondientes y después de completarlos seleccionar el botón Aceptar.
Resultado esperado: Se adiciona una pregunta del tipo selección única correctamente.
Evaluación de la prueba: Satisfactoria

Tabla 34. CPA Adicionar preguntas.

Caso de prueba de aceptación
Historia de usuario: Adicionar preguntas.
Nombre: Adicionar preguntas.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución:
Entradas/Pasos de ejecución: Seleccionar la opción del menú Adicionar preguntas, se selecciona el tipo selección múltiple, aparece un formulario con los campos correspondientes y después de completarlos seleccionar el botón Aceptar.
Resultado esperado: Se adiciona una pregunta del tipo selección múltiple

correctamente.
Evaluación de la prueba: Satisfactoria

Iteración 3:

Tabla 35. CPA Guardar cambios.

Caso de prueba de aceptación
Historia de usuario: Guardar cambios.
Nombre: Guardar cambios.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Debe haber un mínimo de diez preguntas en el fichero.
Entradas/Pasos de ejecución: Después de haber modificado las preguntas deseadas seleccionar la opción del menú: guardar y luego seleccionar el botón Aceptar.
Resultado esperado: Se guardan los cambios realizados correctamente.
Evaluación de la prueba: Satisfactoria

Tabla 36. CPA Guardar cambios.

Caso de prueba de aceptación
Historia de usuario: Guardar cambios.
Nombre: Guardar cambios.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Debe haber un mínimo de diez preguntas en el fichero.
Entradas/Pasos de ejecución: Después de haber modificado las preguntas deseadas seleccionar la opción del menú: guardar como, aparece una ventana para

que escriba el nombre que le va a asignar al fichero y la ruta donde desea guardarlo y luego seleccionar el botón Aceptar.
Resultado esperado: Se guarda el fichero con el nombre especificado y los cambios realizados correctamente.
Evaluación de la prueba: Satisfactoria

Tabla 37. CPA Guardar cambios.

Caso de prueba de aceptación
Historia de usuario: Guardar cambios.
Nombre: Guardar cambios.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Debe haber un mínimo de diez preguntas en el fichero.
Entradas/Pasos de ejecución: Después de haber modificado las preguntas deseadas seleccionar la opción del menú: guardar y luego seleccionar el botón Aceptar.
Resultado esperado: Se guarda el fichero encriptado.
Evaluación de la prueba: Satisfactoria

Tabla 38. CPA Guardar cambios.

Caso de prueba de aceptación
Historia de usuario: Guardar cambios.
Nombre: Guardar cambios.
Responsable: Marlon Rodríguez Mendosa
Condiciones de ejecución: Debe haber un mínimo de diez preguntas en el fichero.

Entradas/Pasos de ejecución: Después de haber modificado las preguntas deseadas seleccionar la opción del menú: guardar como, aparece una ventana para que escriba el nombre que le va a asignar al fichero y la ruta donde desea guardarlo y luego seleccionar el botón Aceptar.
Resultado esperado: Se guarda el fichero encriptado.
Evaluación de la prueba: Satisfactoria

Resultados de las pruebas de aceptación

Durante el proceso de pruebas, se realizaron un total de 12 casos de pruebas de aceptación. En la ejecución de dichas pruebas a la aplicación se realizaron 3 iteraciones, en las cuales se solucionaron las no conformidades detectadas. En la primera iteración no se detectaron no conformidades, luego en la segunda iteración surgieron 4 no conformidades y en la tercera iteración no se obtuvieron no conformidades por lo que fueron evaluados todos los casos de prueba de satisfactorio. En la Ilustración 3 se reflejan estos resultados.

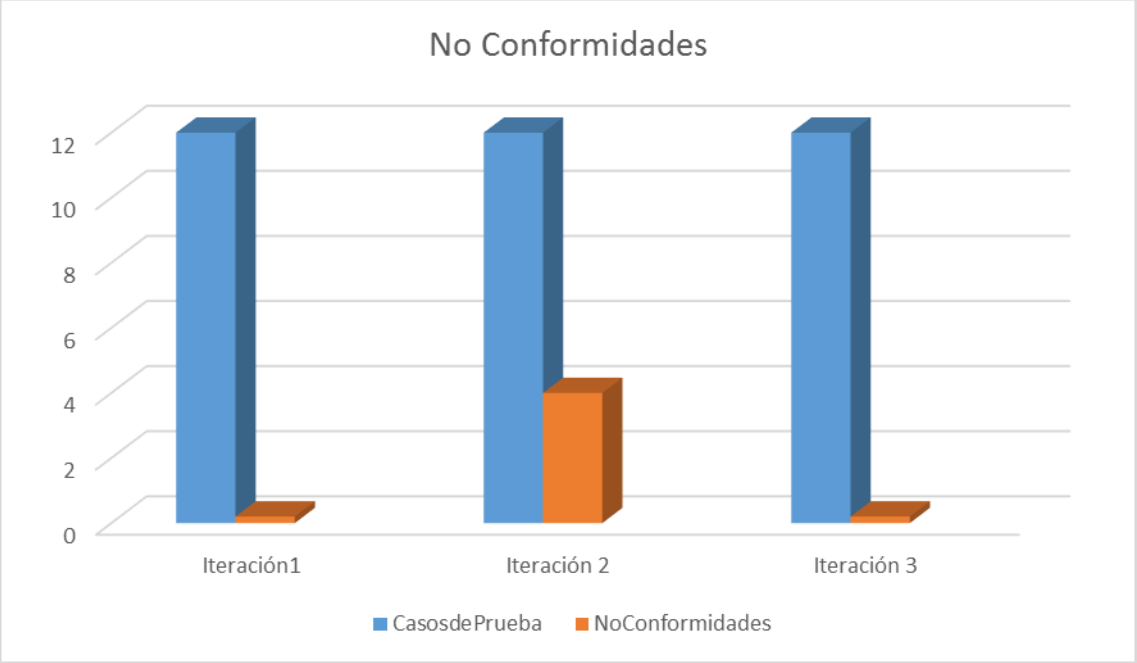

Ilustración 3. Gráfico de no conformidades.

3.4 Consideraciones parciales del capítulo

En el capítulo se realizó la construcción de una versión de la solución propuesta que corresponde con el 100% de la aplicación final. Además se desarrollaron las fases de implementación y prueba de la metodología XP. Para la comprobación del funcionamiento del *software* se realizaron pruebas de aceptación donde luego de la tercera iteración se obtuvo un 100% de satisfacción.

Conclusiones

Con la implementación de una aplicación capaz de permitir a los profesores de química manipular la edición y protección del fichero de preguntas teóricas en el LVQ, se dio cumplimiento al objetivo de la investigación presentada. Se demostró que se puede modificar el contenido de una aplicación sin la necesidad de recompilarla lo que convierte al usuario final en parte del equipo de desarrollo. También se mostró que es posible proteger ficheros utilizando algoritmos de encriptación programados directamente en el código sin utilizar códigos ajenos que puedan propiciar comportamientos indeseados por parte del sistema.

Referencias Bibliográficas

1.Pere Marquès, Universidad Autónoma de Barcelona. http://www.lmi.ub.es/te/any96/marques_software. 2016.

2. [En línea] http://www.gestiopolis.com/uso-del-software-educativo-en-el-proceso-de-ensenanza-y-.

3. [En línea] http://dle.rae.es.

4. [En línea] http://www.definicionabc.com/comunicacion/editor.php..

5. [En línea] http://www.enciclopediadetareas.net/2011/12/editores-graficos.html.

6. [En línea] http://revista.ibict.br/ciinf/index.php/ciinf/article/view/745/619.

7. [En línea] http:www.w3c.org/XML.

8. [En línea] http://www.w3schools.com/xml/xml_whatis.asp.

9. [En línea] http://raulespinola.wordpress.com/2009/03/27/criptografia-y-la-informatica/.

10. [En línea] https://www.icai.es/contenidos/publicaciones/anales_get.php?id=1210.

11. [En línea] http://www.segu-info.com.ar/proyectos/p1_algoritmos-basicos.htm.

12. [En línea] http://comunidad.dragonjar.org/f156/lista-de-algoritmos-de-encriptacion-8807/.

13. [En linea] http://wwwdi.ujaen.es/es/mlucena/lcripto.html.

14. Universidad Pontificia Comillas. *Introducción a la criptografía. Tipos de algoritmos. .* Escuela Técnica Superior de Ingeniería (ICAI) : s.n.

15. Pressman, Roger. *Ingeniería de Software: Un enfoque práctico, 7ma edición.*

16. Figueroa Roberth, Solís Camilo, Cabrera Armando. *Metodologías tradicionales y metodologías ágiles.* (2007).

17. Letelier, Patricio, Canós, José H. & Penadés, Mª Carmen. *Metodologías ágiles en el desarrollo de software: Extreme Programming (XP).* 2003.

18. Mendoza Sánchez, María A. *Metodologías del desarrollo de software.* 2004.

19. Flores, Lic. Ervin. *Metodología agiles Proceso Unificado Ágil (AUP)*. (2006).

20. Schwaber, K. *Advanced Development Methods. SCRUM Development .* 1ro de Julio del 2010.

21. Kniberg, Henrik. *Scrum y XP desde las trincheras.pdf.* 2007.

22. **Loucopulos, P, Karacostas, V.** ***System Requirements engineerinuality which will perform effectively.*** **1995.**

23. Visual Paradigm. [En línea] 2012. http://www.visual-paradigm.com./.

24. [En línea] https://www.qt.io.

25. **JOSKOWICZ, J.** ***Reglas y prácticas en eXtreme Programming. .*** **Universidad de Vigo : s.n., 2008.**

26. **ACOSTA, R. V. y PIS, V. M. A.** ***Portal web para la gestión de información de los procesos sustantivos de la Facultad 5 de la Universidad de las Ciencias Informáticas.*** **2013.**

27. **Panadés, Patricio Letelier y María del Carmén.** ***Metodologías Ágiles para el desarrollo de software.*** **Departamento de Sistemas Informáticos y Computación, Universidad Politécnica : s.n., 2012.**

28. ***Desarrollo ágil de software aplicando programación ,Revista Ingenio.*** **GÓMEZ, A. R. y DUARTE, A. Q.** 2014.

29. **Isidro Ramos Salavert, María Dolores Lozano Pérez.** ***Ingeniería Del Software Y Bases de Datos: Tendencias Actuales.*** **2000.**

30. **Atuesta, Claudia Maria Zea Restrepo y Maria del Rosario.** ***Hacia Una .*** **Medellin : s.n., 2007. (1)**

31. **[En línea] https://revistatelematica.cujae.edu.cu/idex.php/tele/article/viewfile/15/10.**

32. Developer Network. Developer Network. [En línea] 2015. https://msdn.microsoft.com/es-es/library/bb972240.aspx.

33. **Addison Wesley**, **Patterns, Design.** ***Elements of Reusable Object-Oriented Software.*** **ISBN 978-0201633610. 1994.**

34. SINGLETON. [En línea] 5 de junio de 2015. https://msdn.microsoft.com/es-es/library/bb972272.aspx.

35. PRUEBAS. [En línea] 2015. http://www.lsi.us.es/~javierj/investigacion_ficheros/PSISEXTREMA.pdf.

36. **CORBEA, M. R. y PÉREZ, M. O.** ***LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA. TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS .*** **Universidad de las Ciencias Informáticas : s.n., 2007.**

Tabla 39 Tarjeta CRC – Widget

Tarjeta CRC	
Clase: Widget	
Responsabilidades: Procesar los slot processQuestion y processEliminar, convierte al parámetro en una cadena seguida por el código que representa a la función que se desea ejecutar.	**Colaboradores:** BusinnessManager XmlManager MainWindow

Tabla 40 Tarjeta CRC – WidgetTrueorFalse

Tarjeta CRC	
Clase: WidgetTrueorFalse	
Responsabilidades: Construir un widget del tipo de ejercicio verdadero o falso	**Colaboradores:** ExercisesToF BusinnessManager XmlManager MainWindow

Tabla 41 Tarjeta CRC – WidgetSeleccionMultiple

Tarjeta CRC	
Clase: WidgetSeleccionMultiple	
Responsabilidades: Construir un widget del tipo de ejercicio	**Colaboradores:** ExercisesSM

selección múltiple	BusinnessManager XmlManager MainWindow

Tabla 42 Tarjeta CRC – WidgetSeleccionOne

Tarjeta CRC	
Clase: WidgetSeleccionOne	
Responsabilidades: Construir un widget del tipo de ejercicio selección única	**Colaboradores:** ExercisesSU BusinnessManager XmlManager MainWindow

Tabla 43 Tarjeta CRC – BusinnessManager

Tarjeta CRC	
Clase: BusinnessManager	
Responsabilidades: Posee una sola instancia de la clase controladora para acceder a los datos y modificarlos	**Colaboradores:** XmlManager MainWindow

Tabla 44 Tarjeta CRC – AddExerciseWindow

Tarjeta CRC	
Clase: AddExerciseWindow	
Responsabilidades: Mostrar el formulario de adicionar ejercicios y	**Colaboradores:**

capturar los datos	ExercisesSM ExercisesToF ExercisesSU BusinnessManager XmlManager MainWindow