Universidad de las Ciencias Informáticas

Facultad 2

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE

INGENIERO EN CIENCIAS INFORMÁTICAS

Herramienta para la Evaluación Automatizada de expresiones del lenguaje Cálculo Relacional

Autores
Damaysis Carmona Hernández
Alexei Pupo Ricardo

Tutores
Lic. Yamilka Gómez León, Asistente
Msc. Ailec Granda Dihigo, Profesor Auxiliar
Dr.C. Edistio Yoel Verdecia Martínez, Profesor Titular

"Año 55 de la Revolución"

La Habana, Cuba, Junio, 2013

“Para ese mundo futuro con el que ustedes tendrán que lidiar, los conocimientos tienen una importancia trascendental.”

Fidel Castro

DECLARACIÓN DE AUTORÍA

Declaro que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas y a la Facultad 2 para que hagan el uso que estimen pertinente de este trabajo.

Para que así conste firmamos la presente a los ____ días del mes de junio del 2013.

Autores

_____ _____

Damaysis Carmona Hernández Alexei Pupo Ricardo

Tutores

_____ _____

Lic. Yamilka Gómez León Msc. Ailec Granda Dihigo

_____

Dr.C. Edistio Yoel Verdecia Martínez

AGRADECIMIENTOS

Damaysis:

A mis padres Marisol y Danilo, que los adoro, gracias por confiar en mí en todo momento, por apoyarme en los momentos más difíciles y sobre todo por comprenderme siempre.

A mi hermano, que a pesar de ser pequeño, ha sabido entenderme y ofrecerme su apoyo.

A mi abuela Isabel por ser mi segunda madre y estar al tanto de todo a mi alrededor.

A mis tutores Edistio Yoel Verdecia Martínez, Ailec Granda Dihigo y Yamilka Gómez León por su gran apoyo y comprensión durante todo este tiempo.

A mi amiga Anabel que siempre ha estado a mi lado, sin importar la gravedad de la situación.

A mi amiga Lidiagnis por su paciencia y solidaridad en todo momento.

A mi amigo Carlos por sus consejos y por su apoyo incondicional.

A mi compañero de tesis Alexei por soportarme en toda esta larga pero fructífera trayectoria.

A mi novio Roberto por preocuparse por mí, por su cariño, paciencia y dedicación constante.

A todas mis amistades y a todas las personas que he conocido a lo largo de mi carrera que a lo largo de la carrera se han preocupado por mí y me han ofrecido su ayuda cuando más lo necesitaba.

Agradezco infinitamente a todos los profesores que de alguna forma colaboraron en la confección de la herramienta, Adrián Maranje, Rainer Segura, Victor Ernesto Marín, Jorge Heriberto, Ariel Díaz, Bárbara Triana, entre otros.

A todo el que hizo posible que mi sueño se materializara y hoy haya terminado satisfactoriamente este trabajo de diploma.

AGRADECIMIENTOS

Alexei:

A mi madre bella, por ser mi motivo, mi razón, mi vida y la fuerza impulsora para lograr alcanzar este sueño.

A mi hermano, por ser más que un amigo, más que un hermano; por darme su apoyo y siempre estar cuando más lo he necesitado.

A mi padrastro Frank, que siempre ha sido más que un padre para mí.

A mi abuela, por apoyarme siempre y darme todo su amor y cariño.

A mi padre, por su preocupación y por su apoyo durante todo este tiempo.

A mi prima Bertha, a mi tío Ricardo, a mis primos Daya y Ricar, a mi tía Ruby, a toda mi familia que de alguna forma u otra me apoyaron, ayudaron y siempre me brindaron su confianza.

A mis tutores Yamilka, Ailec y Edistio, por su apoyo incondicional durante la elaboración de esta tesis y por soportar tantas molestias causadas.

A Mady, por compartir a mi lado gran parte de mi vida en la universidad y por todo el amor que me dio.

A mi compañera de tesis Damaysis, por su ayuda en los momentos difíciles, por su comprensión, por ser tan buen amiga y por todo el sacrificio realizado en esta etapa.

A todos mis amigos, a Carlos, Anabel, Adnier, Lidiagnis, Jorge Heriberto, a mi gente del tenis, y a todas esas personas que conocí durante esta etapa y me brindaron su ayuda.

A todos los profesores que de una forma u otra colaboraron con la elaboración de esta tesis y a aquellos que contribuyeron a mi formación durante los 5 años en la universidad.

A todos ustedes, mis más sinceros agradecimientos.

DEDICATORIA

Damaysis:

Dedico este trabajo de diploma a mis padres, que han sido lo más importante en mi vida, han sido ese motor impulsor que me ha llevado tan lejos.

A mi padre, por guiarme cuando más lo necesitaba y por brindarme sabios consejos para la vida; y a mi madre, por estar a mi lado siempre y ser amiga por sobre todas las cosas.

Alexei:

A mi madre, por creer siempre en mí, por darme su confianza y todo su amor.

A mi familia, por el apoyo que me brindaron.

RESUMEN

La Universidad de las Ciencias Informáticas utiliza en la formación de los futuros ingenieros, las nuevas Tecnologías de la Información y las Comunicaciones. Entre los temas que se imparten dentro del proceso de formación del Ingeniero en Ciencias Informáticas se encuentran las bases de datos, contenido que se imparte en las asignaturas Sistemas de Bases de Datos (I y II). Estas incluyen el Cálculo Relacional, lenguaje de manipulación de datos del Modelo Relacional, basado en una rama de la Lógica Matemática, la Lógica de Predicados. La asimilación de este contenido para los estudiantes es compleja, pues se imparte en poco tiempo y la Universidad utiliza hoy recursos para su enseñanza que no son suficientes para su aprendizaje. Lo anterior está apoyado en la inexistencia de herramientas para que los estudiantes puedan autoevaluar su preparación y aprender a través de los errores que cometen, cuando utilizan una estrategia de prueba y error. A partir de esta situación, nace la idea de desarrollar una herramienta capaz de identificar errores y evaluar las consultas que formulan los estudiantes, con el objetivo de facilitar el proceso de enseñanza aprendizaje de este lenguaje. La herramienta realiza un análisis detallado de las expresiones introducidas, indicando los errores y comprobando mediante la traducción al SQL si el resultado es correcto o no. Para comprobar su funcionamiento se efectuarán pruebas al sistema. Esta investigación forma parte de un proyecto de maestría que busca definir un conjunto de herramientas para la enseñanza de las asignaturas de bases de datos.

Índice

Índice de las Figuras

Índice de las Tablas

Introducción

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha mostrado un progreso acelerado en las últimas décadas, lo cual ha posibilitado un incremento de su uso y que su aplicación llegue a todos los ámbitos de la sociedad. *Las TIC constituyen un medio que ofrece un acceso instantáneo a la información y a cada persona le corresponde gestionar, enriquecer y construir su conocimiento a partir de las mismas* (1).

A pesar de que Cuba es un país subdesarrollado, se han alcanzado altos niveles de desarrollo social, los cuales han estado enfocados al acercamiento de la población a los avances de las nuevas tecnologías. Entre las acciones desarrolladas se debe mencionar la incorporación de más de 3 millones de personas en cursos de diferente naturaleza, teniendo como objetivo proporcionar una cultura informática a la comunidad cubana. Otra acción del país ha sido el empleo y distribución de computadoras en todos los centros educacionales, gracias a lo cual los estudiantes de todos los niveles y tipos de enseñanza tienen acceso diariamente a múltiples materiales e información útil a través de computadoras instaladas en todas las instalaciones de educación. (2)

Las TIC cobran vital importancia en el sector educacional, ya que tienen gran impacto en la organización y desarrollo del Proceso de Enseñanza-Aprendizaje (PEA). Actualmente, la educación en el mundo enfrenta el desafío del uso de las TIC como herramientas de apoyo al PEA, con el fin de desarrollar estrategias en los estudiantes que les sirvan para enfrentar y solucionar las necesidades de la sociedad en que viven. La incorporación de las TIC a la educación exige pensar previamente cuáles son los objetivos, los retos de la educación, y determinar posteriormente de qué manera y en qué condiciones la presencia de estas tecnologías en las escuelas contribuye a ello. (3)

En Cuba se le ha dado gran importancia a las TIC en todos los niveles de enseñanza, principalmente en la educación superior. La Universidad de las Ciencias Informáticas (UCI), creada el 23 de Septiembre del 2002, desarrolla desde esa fecha actividades académicas y productivas. Tiene como misión formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, y producir aplicaciones y servicios informáticos, a partir de desarrollar el vínculo estudio-trabajo como modelo de formación y servir de soporte a la Industria Cubana del Software. (4)

El proceso de formación de los estudiantes en la UCI, está apoyado por una infraestructura creada para la teleformación, a partir de la utilización del Entorno Virtual de Enseñanza

Aprendizaje Moodle (EVEA UCI) (5).

En esta plataforma se han diseñado cursos que responden a las necesidades de las asignaturas que conforman el plan de estudio de la Ingeniería en Ciencias Informáticas (ICI). Además de lo anterior, los profesores pueden implementar cursos semipresenciales, incluir en los distintos tipos de cursos recursos de aprendizaje y para la autoevaluación, disponiendo los estudiantes de un medio del cual pueden obtener, utilizar o compartir materiales didácticos. (6)

A partir del segundo semestre de segundo año, después de haber cursado tres asignaturas de la disciplina Técnicas de Programación de Computadoras (TPC) y la asignatura Introducción a las Ciencias Informáticas, se le da continuidad a la disciplina Ingeniería y Gestión de Software (IGSW). Esta disciplina tiene gran importancia en la formación de los egresados de la UCI, pues abarca el estudio del ciclo de vida de un software, y además contiene las asignaturas Sistema de Base de Datos 1 y 2. (7) . Estas dos asignaturas, en general, persiguen preparar al estudiante en los conceptos básicos relacionados con las bases de datos. (8)

La asignatura Sistemas de Base de Datos 1 (SBD1), se imparte actualmente durante el cuarto semestre de la carrera. (9) Está organizada en dos temas: Diseño de Bases de Datos Relacionales e Implementación de Bases de Datos. (10) Dentro de los contenidos que se imparten en la asignatura SBD1 en la UCI está el Modelo Relacional (MR), el cual constituye un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos. Este modelo es el más utilizado en la actualidad para modelar problemas reales (11); y la mayoría de los Sistemas Gestores de Bases de Datos (SGBD) lo implementan. Dentro de la parte manipulativa de este modelo se incluye el Cálculo Relacional (CR), el cual constituye un lenguaje de manipulación de datos basado en la Lógica de Predicados o Cálculo de predicados de primer orden. Según el tipo de variables que se manejan, existen dos tipos de CR: el Cálculo Relacional de Tuplas (CRT), que emplea variables de tupla que toman como valores las tuplas que pertenecen a una relación; y el Cálculo Relacional de Dominios (CRD), que utiliza variables de dominio que toman valores de los dominios asociados a los atributos de las relaciones. El presente trabajo está dedicado al Cálculo Relacional de Tuplas, por lo que en lo adelante las referencias al Cálculo Relacional son al CRT. (12)

En el curso virtual asociado a la asignatura SBD1 y publicado en el EVEA de la UCI, existen pocos recursos que apoyen el PEA de este contenido, y los que existen no permiten el desarrollo de las habilidades que se requieren. Dentro de los recursos asociados a este contenido se encuentra una teleconferencia filmada en el curso 2009/2010 y un foro de debate general, los cuales no satisfacen el desarrollo de las habilidades que se requieren para el

aprendizaje del CR. Las dificultades antes descritas en el PEA en la UCI, se reflejan en los resultados de las pruebas parciales y finales de la asignatura SBD1 (Ver Anexo 1).

El total de horas clase que se dedican en la asignatura a la enseñanza del CR es cuatro horas, según el plan calendario actual, lo cual no es suficiente para que los estudiantes puedan asimilar el contenido y desarrollar las habilidades necesarias en la formulación de expresiones en este lenguaje. Los profesores que imparten estos contenidos han constatado a partir de su experiencia, que para los estudiantes es difícil reconocer cuándo las consultas expresadas en el papel son correctas y responden a los requisitos de información planteados.

En una encuesta realizada a los profesores de SBD1 (Ver Anexo 2), se puntualizaron temas con respecto al uso de herramientas de evaluación automatizada en la asignatura. A partir de una muestra de diecisiete (17) profesores con cuatro (4) años de experiencia promedio en la impartición de la asignatura, se obtuvo que el 82% de los encuestados coinciden en que las frecuencias que se dedican a este tema no son suficientes, además no brindan al estudiante una retroalimentación adecuada. Esta encuesta arrojó la necesidad de realizar aplicaciones que contribuyan a la enseñanza y aprendizaje del CR con un 70% de los encuestados coincidiendo en este punto, y a su vez de desarrollar herramientas que permitan la evaluación automatizada facilitando así la autopreparación de los estudiantes. Esta automatización en los procesos de evaluación es deseable y va acorde con los avances informáticos y las necesidades de las instituciones por alcanzar uniformidad y más altos niveles educativos.

Una de las dificultades asociadas con SBD1 y con la enseñanza del CR es que no existen herramientas informáticas u otro tipo de recursos que permitan a los estudiantes practicar y autoevaluar la formulación de expresiones. La inexistencia de este tipo de recursos provoca que sea el profesor quien deba revisar de forma manual las expresiones formuladas por los estudiantes, lo que le ocupa un tiempo considerable si se tiene en cuenta que para un mismo enunciado pueden existir múltiples respuestas, que dependen incluso de las variables utilizadas por los estudiantes. La existencia de recursos informáticos que permitan evaluar este tipo de expresiones puede ser una ayuda de inestimable valor para los profesores y estudiantes.

En función de la situación analizada y expuesta se define el **problema a resolver** a partir de la siguiente interrogante: ¿Cómo contribuir a la autoevaluación de los contenidos del Cálculo Relacional, con el apoyo de las Tecnologías de la Información y las Comunicaciones?

El **objeto de estudio** lo constituye la construcción de herramientas para la evaluación automatizada, mientras que el **campo de acción** es la construcción de herramientas para la evaluación automatizada de expresiones del Cálculo Relacional de Tuplas.

Para resolver el problema planteado, se define como **objetivo general**: Desarrollar una herramienta para la evaluación automatizada de expresiones del lenguaje Cálculo Relacional de Tuplas.

Para el desarrollo de la investigación se definieron los siguientes **objetivos específicos:**

1. Establecer las bases teóricas que sustentan la construcción de herramientas informáticas para la evaluación automatizada en las asignaturas de bases de datos.
2. Caracterizar las herramientas existentes para la evaluación automatizada de expresiones del Cálculo Relacional de Tuplas.
3. Definir las herramientas, lenguajes y tecnologías a utilizar en la construcción de la Herramienta para la Evaluación Automatizada de expresiones del Cálculo Relacional de Tuplas (HEA-CRT).
4. Elaborar los artefactos correspondientes con la metodología de desarrollo a utilizar para la elaboración de HEA-CRT.
5. Valorar la calidad de HEA-CRT mediante pruebas de software.

En la presente investigación se utilizan los **métodos** teóricos y empíricos siguientes:

Métodos teóricos

- **Analítico-Sintético**: Se usará para la construcción del marco teórico referencial que sirva de sustento a la utilización de herramientas informáticas en el proceso de enseñanza aprendizaje, con énfasis en el contenido Cálculo Relacional en asignaturas de bases de datos. Además por ser procesos lógicos del pensamiento, serán utilizados en todas las etapas de la investigación.
- **Modelación**: Se crearán modelos para la representación de una manera sencilla del proceso que engloba la interpretación de código en lenguajes del Cálculo Relacional.
- **Sistémico estructural funcional:** Consiste en la elaboración del análisis, diseño e implementación de la herramienta informática que contribuya a mejorar el Proceso de Enseñanza-Aprendizaje del contenido Cálculo Relacional en los estudiantes.

Métodos empíricos

- **Análisis documental, observación científica y encuestas**: Será útil en el análisis de informaciones relacionadas con el funcionamiento de los procesos de interpretación de código en lenguajes del Cálculo Relacional, con el objetivo de extraer elementos importantes que se relacionen con el objeto de estudio y así poder alcanzar conocimientos generalizados sobre las posibles causas del problema de investigación.

Métodos Estadísticos:

- **Estadística descriptiva:** Tienen por objeto fundamental describir y analizar las características de un conjunto de datos, obteniéndose de esa manera conclusiones sobre las características de dicho conjunto y sobre las relaciones existentes con otras poblaciones, a fin de compararlas, Se utilizará en el presente trabajo para el procesamiento y análisis de los resultados obtenidos de las encuestas.

El aporte práctico de esta investigación es el desarrollo de la herramienta HEA-CRT, que permite la evaluación automatizada de expresiones formuladas utilizando el lenguaje CRT. Esta herramienta puede ser utilizada desde el punto de vista didáctico y posibilita que los estudiantes puedan autoevaluar sus conocimientos relacionados con la formulación de consultas utilizando el CRT. HEA-CRT brinda la posibilidad de aprender en un proceso de prueba y error, donde los estudiantes corrijan los errores cometidos, este proceso permite el desarrollo de habilidades y permite sistematizar los contenidos impartidos en las conferencias y clases prácticas. La HEA-CRT para el profesor, se convertirá en un medio de apoyo, que le permitirá utilizar mejor su tiempo y la herramienta les puede brindar una valoración actual de estado y el nivel de conocimiento que poseen sus estudiantes sobre el CRT. La correcta utilización de la herramienta puede contribuir a la obtención de mejores resultados de los estudiantes en las evaluaciones frecuentes, parciales y finales de la asignatura SBD1.

El trabajo está estructurado en introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos. El Capítulo 1**, "Evaluación automatizada de expresiones del Cálculo Relacional, fundamentos teóricos y estado actual"** incluye un estudio del estado del arte asociado a los principales conceptos relacionados con la presente investigación; se abordan además la propuesta de metodología, lenguaje de programación y herramientas de desarrollo a utilizar en la implementación de la solución. El segundo Capítulo **"Análisis y diseño de la herramienta para la evaluación automatizada de expresiones del Cálculo Relacional",** aborda lo referente al análisis y diseño de la HEA-CRT, se especifican las características que el sistema debe cumplir así como el análisis del dominio de la aplicación, y se describen las funcionalidades a automatizar para darle solución al problema. El tercer y último Capítulo **"Implementación y Pruebas de la HEA-CRT",** muestra los resultados del conjunto de pruebas realizadas a la HEA-CRT, además de la descripción del proceso de implementación de la herramienta, que garantizan su correcto funcionamiento.

Capítulo1: Evaluación automatizada de expresiones del Cálculo Relacional, fundamentos teóricos y estado actual

Este capítulo tiene como objetivo presentar un estudio del estado del arte de las herramientas existentes para la evaluación de expresiones del CR, acompañado de la revisión de los principales conceptos asociados al CR y a la evaluación automatizada. Además, en el capítulo se realiza una evaluación de lenguajes de programación, herramientas informáticas, tecnologías y metodologías de desarrollo a utilizar en la elaboración de la propuesta de solución. El capítulo incluye al mismo tiempo los requisitos fijados por el cliente para la elaboración de la propuesta de solución.

1.1 La utilización de las TICs en la educación y la evaluación automatizada

Las TIC han facilitado el acceso a la información y/o al conocimiento, han modificado conceptos como espacio, tiempo e identidad, han redefinido los roles, han cambiado la forma como las personas se comunican, se informan, aprenden e incluso piensan; éstos y otros aspectos no han estado aislados y se han reflejado en la educación.

Las TIC han sido y están siendo incorporadas al proceso educativo desde hace algunos años. Aún no existen estudios concluyentes que permitan afirmar que la utilización de los medios informáticos en la educación ha servido para mejorar los resultados académicos, sin embargo a menudo se refieren a las transformaciones obtenidas en el modo de hacer. Se ha observado que las tecnologías de la información suscitan la colaboración en los alumnos, les ayudan a centrarse en los aprendizajes, mejoran la motivación y el interés, favorecen el espíritu de búsqueda, promueven la integración y estimulan el desarrollo de ciertas habilidades intelectuales tales como el razonamiento, la resolución de problemas, la creatividad y la capacidad de aprender a aprender. (1)

La explotación de las TIC en la docencia universitaria tiene como objetivo principal que los alumnos tengan acceso a los servicios educativos del campus desde cualquier lugar, de manera que puedan desarrollar personal y autónomamente acciones de aprendizaje. (13) En la actualidad existen tecnologías que están principalmente orientadas a la educación y que en la actualidad son motivo de investigación y debate en las universidades; entre ellas se encuentran el e-Learning, los objetos de aprendizaje y los repositorios abiertos de recursos educacionales (14).

Al hablar de e-learning se encontraron términos como LMS (Learning management System) o LCMS (Learning Content Management System) que son plataformas tecnológicas de creación y administración de contenidos; normalmente están compuestas por repositorios, herramientas de autoría, publicación, colaboración, administración e interfaz dinámica (15).

Uno de los elementos que se ha buscado favorecer con la utilización de las TIC en la educación, es el proceso evaluativo. Lo anterior se debe a que se considera que la piedra angular de una formación de calidad lo constituye el uso eficaz de la evaluación.

Se entiende por evaluación, en sentido general, aquel conjunto de procesos sistemáticos de recogida, análisis e interpretación de información válida y fiable, que en comparación con una referencia o criterio permita llegar a una decisión que favorezca la mejora del objeto evaluado, en este caso el aprendizaje del estudiante (16).

La evaluación permite determinar, entre otros aspectos, el nivel de conocimientos que han adquirido y el nivel desarrollo de las habilidades de los estudiantes. Las evaluaciones tienen que ser usadas para guiar al estudiante hacia experiencias de aprendizaje eficaces, confirmando aptitudes, conocimientos y dando motivación a través el sentimiento de realización. Uno de los elementos en este sentido es enseñar a los estudiantes a evaluar su aprendizaje.

En la actualidad, el uso de las TIC en el proceso de evaluación de los estudiantes universitarios, puede constituir un elemento diferenciador respecto a las prácticas evaluativas que hasta ahora se han realizado. (17) Si la evaluación influye tanto en el comportamiento de profesores y alumnos, se debe utilizar dicha influencia para que los resultados del PEA sean los mejores, analizando todos los aspectos del proceso de evaluación en que se pueda influir y que se puedan modificar en el sentido adecuado. (18) Es por ello, que la evaluación automatizada aporta un mayor dinamismo a la retroalimentación y con ello un impacto positivo en el PEA. Por tanto, la evaluación ha de realizarse de manera continuada a lo largo del curso, y las TIC pueden ayudar a lograr este objetivo. Este método de evaluación y estas tecnologías se están convirtiendo en un recurso útil en muchos de ellos, y pueden ayudar en este objetivo de integrar dicha evaluación con la preparación y el desarrollo de las clases.

Además de todas estas consideraciones, existen una serie de peculiaridades que hay que tener en cuenta en procesos de enseñanza aprendizaje que utilicen los entornos virtuales, como son las siguientes: es fundamental evaluar la participación y contrastar si los alumnos han alcanzado determinados aprendizajes y por tanto si han alcanzado los objetivos del curso, es

esencial que el alumno reciba retroalimentación de cómo está siendo su aprovechamiento del curso y las estrategias de evaluación han de ser coherentes con los materiales aportados en clases y en línea, estos se han de presentar con los criterios o referentes de evaluación de forma explícita.

La relevancia otorgada a las TIC en la realización de la evaluación del aprendizaje, tiene en cuenta cada uno de los beneficios que esta aporta. Una de las ventajas es la posibilidad de automatización de este elemento del PEA, positiva tanto para estudiantes como para profesores, por cuanto aporta un mayor dinamismo a la retroalimentación y con ello un impacto positivo en este proceso. Es por ello que surge la evaluación automatizada, aquella que se realiza a través de un sistema informático; con el desarrollo de este tipo de evaluación se puede contribuir a mejorar el proceso de aprendizaje de los estudiantes. (19)

Elementos que reflejan que la evaluación automatizada puede tomar auge a partir de la utilización de las TIC en la evaluación es que se pueden encontrar todo un grupo de programas informáticos que sirven para evaluar el rendimiento de los alumnos a través de Internet, estos se pueden clasificar en tres categorías básicas: (17)

- Entornos virtuales de formación (web-based trining), que ayudan al profesor a gestionar un módulo o curso de enseñanza completo a través de la red (distribución de contenidos, intercambios con los alumnos a través de correo electrónico, foros de discusión o chats y evaluación de los alumnos). Ejemplos de este tipo: WebCT, Learning Space, Edustance, entre otros.
- Herramientas de autor, consisten en software destinado a su vez a la creación de programas a modo de ejercicios o tareas. Ejemplos de este tipo son: Hot Potatoes, Quia!, Clic y algunos más.
- Software específico más complejo (requiere manejo de servidor), que permite poner en la red a toda la institución, tanto para la creación y publicación de los exámenes, como para recoger los resultados de los estudiantes. Ejemplos de este software son: Perception y Quiz Factory.

Utilización de las TIC en el PEA de la UCI

La UCI no ha estado ajena a toda la utilización de las TIC en el PEA, diversas son las propuestas que se pueden encontrar que permiten fundamentar esta afirmación:

- Se desarrolló un simulador para apoyar la visualización de los contenidos del tema Administración de Memoria en la asignatura Sistema Operativo, contenido complejo debido al alto nivel de abstracción que se necesita para asimilarlo e impartirlo. (20)
- Con la intención de mejorar los resultados de la asignatura Teleinformática I se propone la idea de construir una herramienta que sirva de apoyo a los estudiantes y profesores para comprobar los resultados de los ejercicios orientados. (21)
- Se desarrolló un Sistema de gestión del expediente docente en la disciplina Práctica Profesional, lo que permitirá controlar toda la información productiva e investigativa del estudiante de la facultad 10, igualmente permitirá generar la nota de Práctica Profesional a partir del análisis de todas estas actividades. (22)

1.2 La enseñanza de las bases de datos y el Cálculo Relacional

En la actualidad los sistemas de información forman parte de la vida de las personas, su utilización se ha vuelto indispensable. Para los especialistas de la computación y la informática, los encargados de diseñar y construir estos sistemas, tienen gran valor el conocimiento de las bases de datos. En el mundo de la informática el uso de las BD es de vital importancia, ya que se utilizan para almacenar datos que persistan con el paso del tiempo, además pueden considerarse como *un conjunto de datos relacionados entre sí, entendiéndose por dato los hechos conocidos, que pueden registrarse y que tienen un significado implícito* (8). Las bases de datos son necesarias pues facilitan: el almacenamiento de grandes volúmenes de información, apoyan los procesos para la recuperación rápida y flexible de información y la distribución de información en varias formas y formatos. (8)

Hoy en día las BD poseen sistemas para su informatización, que son un conjunto de programas que permite a los usuarios crear y mantener una BD, denominadas SGBD, siendo este un software de propósito general que facilita el proceso de definir, construir y manipular la BD para diversas aplicaciones. (23)

La UCI no está aislada de todos estos avances en la informática, y como parte de la formación de sus ingenieros posee un programa de estudio que cuenta con varias asignaturas que directamente buscan desarrollar las habilidades necesarias para la creación, el manejo y la administración de las bases de datos y los sistemas asociados. Dentro de este grupo de asignaturas se encuentra SBD1, la cual está organizada en dos temas: Diseño de Bases de Datos Relacionales e Implementación de Bases de Datos. Además se pueden mencionar

SBD2, Ingeniería de Software 1, y Programación 4, esta última incluye el tema de acceso a las bases de datos desde aplicaciones web. (10)

Dentro de los contenidos que se imparten en SBD1 se encuentra el Modelo Relacional (MR), el cual constituye un modelo de datos basado en la Lógica de Predicados y en la teoría de conjuntos. Dentro de la parte manipulativa de este modelo se incluye el Cálculo Relacional (CR), el cual constituye un lenguaje manipulación de datos basado en la Lógica de Predicados o Cálculo de Predicados de Primer Orden.

El CR se basa en una rama de la lógica matemática llamada Cálculo de Predicados. Una característica fundamental del CR es la noción de "variable de tupla", también conocida como "variable de rango". Este lenguaje está formado por símbolos (de puntuación, de variables, de constantes, de funciones, de predicados), operadores lógicos y cuantificadores; todos estos agrupados en fórmulas (expresiones) correctas del lenguaje y términos. (24) La importancia del CR radica en el desarrollo de habilidades relacionadas con la formulación de expresiones que permitan recuperar información del esquema relacional correspondiente, lo cual tiene una relación directa con la construcción de instrucciones SELECT del lenguaje SQL. Esta instrucción es la más utilizada en la actualidad para la implementación y trabajo con las bases de datos relacionales. En este proceso se aprende a formular predicados sobre el conjunto de datos que el modelo relacional define, lo que es un elemento básico en el trabajo con bases de datos, pues ellas existen no solo para almacenar información, sino para poder extraer información de ella, y esta extracción en la mayoría de los casos se hace a través de instrucciones de selección.

Existen dos tipos de CR que a continuación serán abordados.

Cálculo Relacional basado en Tuplas (CRT)

El CRT se basa en la especificación de un cierto número de variables de tupla (25). Una variable de tupla es una variable que toma valores sobre una relación, o sea, sus únicos valores permitidos son tuplas de la relación. Si la variable de tupla T toma valores sobre la relación R, entonces en cualquier momento del tiempo T representa a alguna tupla t de R. Se define con la siguiente instrucción: RANGE OF T IS R, siendo T la variable de tupla y R una relación.

Una instrucción de recuperación del Cálculo Relacional tiene la siguiente sintaxis general:

<Lista objeto> [WHERE < predicado>], donde

Lista objeto: especifica qué atributos y de qué relaciones se desean recuperar. Está formada por nombres calificados de atributos separados por comas.

Predicado: especifica las condiciones que deben cumplir las tuplas seleccionadas y es opcional. Puede estar formado por comparaciones entre dos nombres de atributos o entre un nombre de atributo y una constante. Un predicado incluye los **operadores lógicos** para definir una expresión en cualquiera de las formas siguientes:

- (predicado)
- NOT predicado
- predicado AND predicado
- predicado OR predicado
- EXISTS nom-var (predicado)
- FORALL nom-var (predicado)
- o cualquier combinación de las anteriores

Cada ocurrencia de una variable de tupla dentro de un predicado puede estar libre o acotada de manera existencial o universal. Está acotada si esa variable ha sido introducida por un cuantificador EXISTS o FORALL, de lo contrario, la variable es libre. La noción de variable libre es análoga a la de variable global en un lenguaje de programación, o sea, una variable definida fuera del procedimiento actual. Una variable acotada es similar a una variable local, o sea, una definida dentro del procedimiento actual y que no puede ser referenciada desde afuera. Los cuantificadores EXISTS y FORALL del CR juegan el papel de las declaraciones en un lenguaje de programación.

Ejemplo del uso del lenguaje CRT:

Declaración de variables de tuplas:

RANGE OF ED IS edificio

RANGE OF AP IS apartamento

RANGE OF PE IS persona

RANGE OF ES IS estudiante

RANGE OF PR IS profesor

RANGE OF PO IS posta

RANGE OF PP IS posta_persona

Instrucciones de recuperación:

ED.idedificio WHERE EXISTS AP (AP.idedificio = ED.idedificio AND EXISTS PE (PE.idedificio = AP.idedificio and PE.idapto = AP.idapto))

Obtiene el identificador de los edificios que tienen al menos una persona asignada a alguno de sus apartamentos.

AP.idapto, ED.idedificio, PR.categoria WHERE AP.idedificio = ED.idedificio AND EXISTS PE (PE.idedificio = AP.idedificio and PE.idapto = AP.idapto AND PR.idpersona = PE.idpersona)

Obtiene los identificadores de los apartamentos, el identificador del edificio donde están ubicados, y la categoría de los profesores que viven en cada uno, en caso de que existan.

PE.nombre, PE.apellido WHERE EXISTS PP (PP.idpersona = PE.idpersona AND PP.fecha >= '01/01/2013' AND PP.fecha <= '31/12/2013' AND EXISTS PO(PO.idposta = PP.idposta AND PO.idedificio = 96))

Obtiene el nombre y apellido de las personas que tienen al menos una guardia planificada en el año 2013, en el edificio con identificador 96.

Cálculo Relacional basado en Dominios (CRD)

Está constituido por los mismos operadores que el CRT, pero no hay tuplas sino variables de dominio. Las expresiones del cálculo relacional de dominios son de la forma { (x, y, z, ...) / P(x, y, z, ...) }. Donde x, y, z representan las variables de dominio, P representa una fórmula compuesta de átomos (igual que en el CRT). (26)

1.3 Requisitos fijados por el cliente para la propuesta de solución

Como se analizó en los epígrafes precedentes de este capítulo, el estudio de contenidos relacionados con bases de datos y la evaluación automatizada son elementos que tienen complejidades, es por ello que el Departamento Metodológico Central de Ingeniería y Gestión de Software (DMC IGSW) se ha propuesto elaborar un conjunto de herramientas que apoyen el proceso de enseñanza aprendizaje en las asignaturas de Bases de Datos 1 y 2. El desarrollo de la HEA-CRT es parte de un sistema que se desarrolla en común con una similar para el lenguaje Álgebra Relacional (AR), de los autores: Lidiagnis Fonseca Pérez y Adnier Castellanos Puentes.

El desarrollo de la HEA-CRT tiene como cliente al DMC IGSW, el cual requiere de una herramienta con las siguientes características:

1. Utilizar la notación definida por el Departamento Metodológico Central de Ingeniería y Gestión de Software de la UCI.
2. Permitir definir las características de los ejercicios a resolver por los estudiantes.
3. Poder ser ejecutada en múltiples plataformas, considerando el software libre y de código abierto. La universidad y el país hoy realiza un esfuerzo para introducir el Software Libre, por lo que la herramienta no debe ser desarrollada utilizando

lenguajes y herramientas que limiten su ejecución a plataformas propietarias.

4. Traducir del CRT al SQL.
5. Funcionar bajo una arquitectura cliente/servidor.
6. Contener un núcleo que permita la integración al EVEA UCI.
7. Estar disponible 24x7x365 para que los estudiantes la puedan utilizar asincrónicamente.
8. Utilizar PostgreSQL como SGBD, definido para el proyecto por el departamento.
9. Contar con una interfaz gráfica que permita retroalimentar al estudiante de los errores cometidos.

1.4 Traductor

El objetivo principal de esta investigación es desarrollar una herramienta para la evaluación automatizada de expresiones del lenguaje Cálculo Relacional de Tuplas, realizando una traducción de estas expresiones al SQL. Esta traducción será utilizada para evaluar la rectitud de la solución dada por el estudiante, facilitando así el proceso de evaluación automatizada del aprendizaje. Para ello es necesario estudiar los principales conceptos asociados a un traductor entre lenguajes, que consiste en un programa que toma como entrada un programa escrito en un lenguaje de programación (lenguaje fuente) y produce como salida un programa en otro lenguaje (lenguaje objeto). El traductor se escribe en un lenguaje denominado lenguaje de implementación. (27)

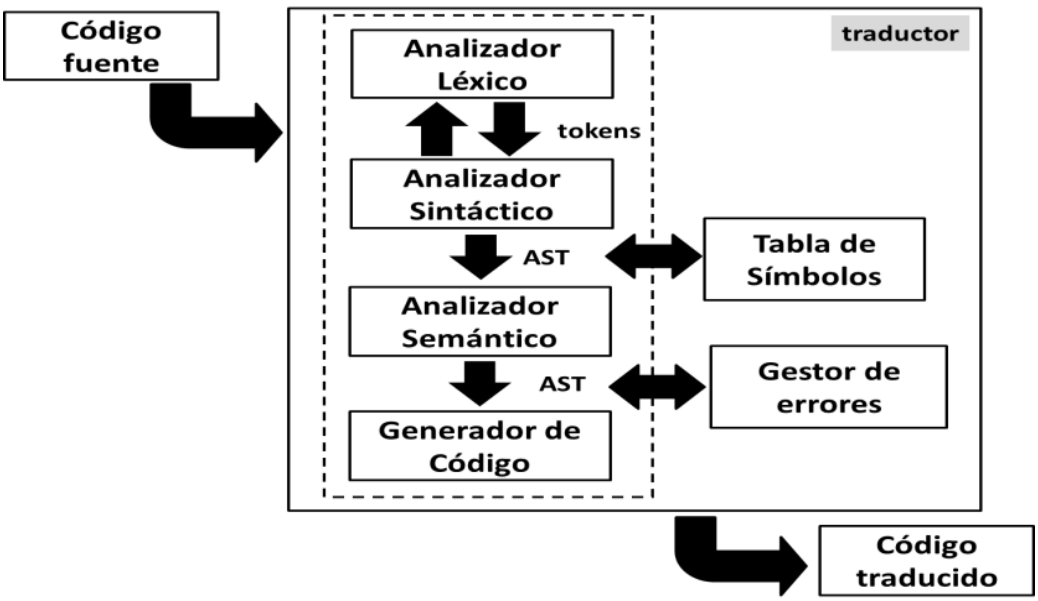

Figura 1.1 Esquema de un traductor (elaboración propia)

El proceso de traducción se compone internamente de varias fases, que realizan distintas operaciones. Cada una de estas fases funciona como piezas separadas dentro del traductor, y pueden escribirse como módulos codificados separadamente aunque en la práctica a menudo se integren y tienen elementos que los componen.

La primera fase de un traductor se denomina **análisis léxico** (scanner[1])**,** su tarea consiste en dividir la entrada en una serie de componente léxicos, realizando en cada uno determinadas acciones. Algunas de estas acciones son: comprobar alguna restricción adicional (por ejemplo que el valor de un literal entero esté dentro de un rango), preparar los atributos del componente y emitir u omitir dicho componente. Así pues, la especificación del analizador léxico deberá incluir por cada categoría léxica del lenguaje el conjunto de atributos y acciones asociadas. (28)

La fase siguiente se denomina **análisis sintáctico** (parser), que constituye el proceso en el cual se examina una secuencia de tokens[2] para determinar si el orden de esa secuencia es correcto de acuerdo a la definición sintáctica del lenguaje. La entrada del analizador sintáctico es la secuencia de tokens generada por el analizador léxico y la salida es la muestra del cumplimiento de las reglas gramaticales que definen al lenguaje. La función principal de la fase de Análisis Sintáctico es verificar que el programa de entrada sea válido, construyendo y devolviendo el Árbol de Sintaxis Abstracta (AST). (27)

La siguiente fase se denomina **análisis semántico** (checker). El analizador semántico es el encargado de detectar la validez semántica de las sentencias aceptadas por el analizador sintáctico. El analizador semántico recibe la información resultado del análisis sintáctico que generalmente se representa utilizando un árbol con la información relativa a la organización jerárquica gramatical de los tokens en la instrucción que se analiza. (27)

El generador de código transforma la salida del análisis semántico en código escrito en el lenguaje a traducir. En un traductor a esta última fase se le denomina ***generación de código***.

1.5 Herramientas para la evaluación de expresiones del Cálculo Relacional.

En la actualidad, el CR es usado ampliamente con fines docentes, es por ello que no se encontraron herramientas desarrolladas para otros fines.

Para la caracterización de las herramientas encontradas se utilizaron los siguientes indicadores:

1- Origen (Internacional (Int) o Nacional (N)).

1 Un analizador *léxico* o analizador lexicográfico (en inglés *scanner*) es la primera fase de un compilador.

2 Conjunto de lexemas que puede ser tratado como una unidad sintáctica.

2 Conjunto de lexemas que puede ser tratado como una unidad sintáctica.

2- Lenguajes de datos con los que trabaja (AR/CRT/CRD).
3- Utiliza notación matemática habitual (Sí/No).
4- Realiza traducción de las expresiones al SQL (Sí/No).
5- Ejecuta las consultas (Sí/No).
6- Permite la gestión de bases de datos (Sí/No).
7- Incluye opciones de optimización de consultas (Sí/No).
8- Utiliza árboles o diagramas para la representación de las consultas (Sí/No).
9- Permite la conexión a algún SGBD (Sí/No).
10- Detecta errores en el planteamiento de las consultas (Sí/No).
11- Realiza corrección automatizada de errores en las consultas (Sí/No).
12- Su distribución es gratuita (Sí/No).
13- Es multiplataforma (Sí/No).
14- Lenguaje o lenguajes en los que fue desarrollada.
15- Tipo de interfaz (Gráfica/Comandos).

Además se incluyen los datos relacionadas con la universidad donde fue desarrollada, y el proyecto al que está vinculado.

Windows Relational Database Interpreter WinRDBI

Fue desarrollada por la Arizona State University. Su principal objetivo es apoyar el PEA de los lenguajes relacionales (CR y AR) mediante la práctica de los estudiantes. Su última versión está desarrollada en el lenguaje de programación Java y utiliza Amzi! Prolog para la interpretación de consultas. Funciona tanto para Linux como para Windows. WinRDBI es un componente integral utilizado para lograr la comprensión de las capacidades que poseen los lenguajes de consulta para BD relacionales, entre ellos el AR, el CRT, el CRD y el SQL en su versión SQL-92. Tiene una interfaz de usuario amigable y completo, mediante la que se pueden crear y cargar BD relacionales desde diferentes formatos de archivos, insertar información en ellas y formular y guardar consultas en los leguajes mencionados. La herramienta incluye una retroalimentación inmediata a los estudiantes pues visualiza las soluciones a las consultas propuestas.

Una de sus desventajas es que no realiza correcciones automatizadas a las soluciones propuestas, por lo que la retroalimentación no es completa para los estudiantes. Adicionalmente, debido a que fue desarrollado sobre la base del establecimiento de tecnología de BD deductivas, que utilizan un lenguaje lógico para consultar las instancias de las BD almacenadas como hechos lógicos y los estudiantes tienen que aprender otro lenguaje. (29)

Web dinámica para el aprendizaje del CRT.

La herramienta fue desarrollada en la Universidad de Burgos. Es una página web dinámica y tiene un carácter multiusuario. Su funcionalidad principal está encaminada a la traducción de expresiones formuladas en CRT a SQL. También permite mostrar el resultado de la ejecución de la consulta sobre una BD centralizada y controlada por el profesor.

Su compilador fue construido utilizando la herramienta JavaCC. Traduce una expresión de CRT a SQL, previo análisis léxico, sintáctico y semántico. Durante este proceso también construye un árbol sintáctico, maneja errores y utiliza una tabla de símbolos para comprobar la validez de la consulta. Interactúa con el SGBD PostgreSQL, a través de un driver JDBC. No obstante su diseño permite una fácil integración con otros SGBD que utilicen el estándar SQL92. Su interfaz es sencilla, a través de un navegador convencional, sin requerimientos de software adicionales. Dentro de sus desventajas se encuentra que no utiliza la notación matemática habitual, no incluye opciones de optimización de consultas y no incluye ninguna forma de representación gráfica de estas. Además tampoco realiza corrección automatizada. (30)

RelationalQuery.

Esta aplicación fue desarrollada en la Universidad de Sevilla, bajo la concepción de software libre. Puede descargarse de su sitio web y se incluye su código fuente. Según su licencia, tanto la herramienta como sus componentes pueden reutilizarse en proyectos libres o propietarios. RelationalQuery fue desarrollada en Java. Su interfaz gráfica se encuentra solamente en idioma inglés aunque, debido a que los mensajes del sistema se almacenan en ficheros XML, se pueden modificar fácilmente. Incluye una interfaz mediante líneas de comando y no permite el uso de la notación matemática habitual.

Dentro de sus funcionalidades se incluye la traducción de expresiones en AR o CRT a consultas SQL. Permite almacenar y recuperar consultas en AR, CRT o SQL. Las consultas se pueden hacer directamente sobre una BD en un SGBD, mediante la utilización de un JDBC.
Esta aplicación no incluye opciones para la optimización de consultas ni su representación en diagramas o árboles. Una de sus debilidades es que es un proyecto que no ha tenido seguimiento. Utiliza WinRDBI para la ejecución de las consultas, por lo que los estudiantes tienen que aprender otro lenguaje. (30)

AR y CR con un enfoque de programación.

Fue desarrolla en la Weber State University, estado de Utah, Estados Unidos. El desarrollo de las expresiones no se concibe en términos matemáticos ni con la notación matemática usual,

sino con un enfoque a la programación. Se considera que esta manera la representación es más familiar para los estudiantes que la representación matemática. La versión de esta librería fue desarrollada en FoxPro y nombrada RALGPROC. Cada función tiene uno o dos parámetros de entrada que son tablas, y se pueden adicionar otros en dependencia de las características de la operación a realizar por la función.

Las expresiones en CR se formulan a partir de la definición de funciones que representan predicados, en las que los parámetros constituyen filas de las tablas, o atributos. Se implementan con programas en Turbo Prolog, donde cada tabla y consulta es representada por un predicado. El procesador de consultas en Prolog para dar la solución, busca en los predicados de tabla para determinar cuáles valores de filas y columnas provocan que el predicado de la consulta sea verdadero. Esta herramienta no es web. Su funcionalidad es exclusivamente mostrar la solución de las consultas que se formulan. La manera en que son implementados los lenguajes impide que se pueda utilizar la notación matemática habitual. La concepción para la formulación de las expresiones implica que se pierda la esencia de estos lenguajes. (30)

Herramientas existentes en el ámbito nacional para el CR

En el proceso de búsqueda de información en la web y en las diferentes universidades del país, no se encontraron herramientas similares a las expuestas anteriormente que contribuyan al proceso de evaluación del aprendizaje en CR.

La siguiente tabla muestra el resumen de las herramientas analizadas. En la misma no se incluye el primer indicador, pues todas las herramientas tienen un origen internacional.

Herr.	Indicador													
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
2.1	AR CRT CRD	No	Sí	Sí	Sí	No	No	Sí	Sí	No	Sí	Sí	Java y Amzi! Prolog	Gráfica
2.2	CRT	No	Sí	Sí	Sí	No	No	Sí	Sí	No	-	Sí	Java	Gráfica
2.3	AR CRT	No	Sí	Sí	No	No	No	Sí	Sí	No	Sí	Sí	Java	Gráfica Comandos
2.4	AR CRT	No	No	Sí	No	No	No	-	No	No	-	-	Visual FoxPro	-

Tabla 1.1 Herramientas para la evaluación de expresiones del Cálculo Relacional.

Leyenda: Herramienta por cada fila de la tabla:

2.1- Windows Relational Database Interpreter: WinRDBI.

2.2- Web dinámica para el aprendizaje del CR.

2.3- Relational Query.

2.4- AR y CR con un enfoque de programación.

Las herramientas antes descritas fueron encontradas en el ámbito internacional, esa es la razón por la cual el indicador no se incluye en la tabla anterior. Todas fueron desarrolladas para el lenguaje CR pero no utilizan la notación matemática habitual y en la UCI no se pueden usar porque se utiliza una notación propia. Todas ellas ejecutan las consultas introducidas por el usuario en la aplicación e incluyen opciones de optimización de consultas, mostrando mediante árboles o diagramas la representación de las consultas y realizan la corrección automatizada de errores a dichas consultas. La gran mayoría son sistemas multiplataforma, realizan traducción de las expresiones al SQL, permiten además la conexión con algún SGBD y detectan errores en el planteamiento de las consultas. Utilizan Java como lenguaje de programación y las gráficas como tipo de interfaz.

1.6 Propuesta de solución

Luego del análisis de los requerimientos planteados por el departamento y de realizada una búsqueda de herramientas similares, se tiene como propuesta de solución elaborar una herramienta que realice la traducción del lenguaje CRT al lenguaje SQL, verificando su solución a partir de la ejecución y comparación de los resultados con una consulta SQL equivalente y correcta. En esta herramienta el profesor definirá los ejercicios a resolver por los estudiantes. Para dar cumplimiento a este objetivo la aplicación contará con las siguientes funcionalidades, relacionadas con la interacción de estudiantes y profesores:

- Conectar a la BD (IP servidor, usuario, contraseña, nombre de la BD).
- Gestionar ejercicios, esto incluye la adición, modificación y eliminación.
- Validar léxica, sintáctica y semánticamente una solución introducida por los estudiantes.
- Traducir solución del lenguaje Cálculo Relacional a SQL.
- Comprobar y retroalimentar la solución del estudiante.
- Mostrar resultados de las consultas.

Para la implementación de la herramienta se hace necesario realizar un estudio de las herramientas, los lenguajes de programación y las tecnologías a utilizar en la elaboración de esta aplicación.

1.7 Definición de la metodología, lenguajes y otras herramientas para el desarrollo de la solución

Este traductor es un software, por lo que su proceso de desarrollo es similar al de otros sistemas de software más complejos. El proceso de desarrollo de un software es definido por Letelier, como un proceso complejo donde: *las personas desempeñan uno o más roles específicos, utilizan diferentes herramientas para producir artefactos y ejecutan diferentes actividades. El avance del proyecto en el tiempo es controlado mediante hitos que establecen un determinado estado para ciertos artefactos. Los artefactos son resultados obtenidos en cada una de las fases del proceso* (31), esta definición coincide con lo planteado por diferentes autores como Jacobson (32) y Pressman (33).

Es por ello que cuando se decide la construcción de un software hay que seleccionar la metodología de desarrollo, el sistema de gestión de bases de datos, los lenguajes de programación y las herramientas que permitirán desarrollarlo.

Desarrollar un buen software depende de un sin número de actividades y etapas, donde el impacto de elegir la mejor metodología para un equipo en un determinado proyecto, es trascendental para el éxito del producto. El papel preponderante de las metodologías es sin duda esencial en un proyecto y en el paso inicial, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo.

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos (software). Se clasifican en dos tipos de metodologías, las ágiles, ejemplo: la Programación Extrema (XP), Scrum; y las pesadas, ejemplo: el Proceso Unificado de Rational (RUP); siendo estas las más usadas de cada tipo. Actualmente no existe una metodología de desarrollo de software que sea global, es decir, que encierre características que puedan aplicarse a cualquier tipo de proyecto. Las características de cada proyecto conjuntamente con su equipo de desarrollo, recursos, y requisitos exigen que se escoja una que se adapte en la mayor medida posible a estas características. (34)

A continuación una breve comparación entre los diferentes tipos de metodologías:

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios

Metodologías Tradicionales	Metodologías Ágiles
	durante el proyecto
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios.
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Más artefactos	Pocos artefactos
Más roles	Pocos roles
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible

Tabla 1.2 Comparación entre metodologías ágiles y pesadas.

Diferencias por las características del Proyecto

Modelo de Proceso	Tamaño del Proceso	Tamaño del Equipo	Complejidad del Problema
RUP	Medio / Extenso	Medio / Extenso	Medio / Alto
XP	Pequeño / Medio	Pequeño	Medio / Alto

Tabla 1.3 Diferencia entre metodologías, por las características del proyecto.

En esta tabla se presenta una comparativa de los modelos de proceso en cuanto a las características del proyecto, analizándose el tamaño del proceso, del equipo y la complejidad del problema para cada uno de los modelos. Se puede resaltar que: con un pequeño equipo de desarrollo se pueden realizar grandes proyectos, de alta complejidad; es por ello que se decidió emplear la metodología ágil XP. (34)

Extreme Programming XP, es una metodología ágil para el desarrollo de software que consiste básicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lograr un producto de buena calidad en poco tiempo. XP está

diseñada para entornos dinámicos y se basa en el trabajo orientado directamente al objetivo, utilizando las relaciones interpersonales como clave para el éxito en el desarrollo de software. La metodología está pensada para equipos de hasta 10 programadores y orientada fuertemente hacia la codificación con énfasis en la comunicación informal. (35)

Esta metodología se basa en una realimentación continua entre el cliente y el equipo de desarrollo con una comunicación fluida entre todos los participantes; también busca simplificar las soluciones implementadas para múltiples cambios. Este tipo de programación es la adecuada para los proyectos con requisitos imprecisos, muy cambiantes y con un riesgo técnico excesivo. (34)

Una característica importante de XP, es que el código siempre se produce en parejas, que van cambiando constantemente para lograr así que todo el equipo sepa y pueda modificar según necesidades el código generado. Esto logra en el equipo que los integrantes aprendan entre sí y compartan todo el código, característica por la que también se utilizará dicha metodología.

Uno de los elementos más importantes a elegir a la hora de desarrollar de un software, es el lenguaje en que se comunicaran los diferentes miembros del equipo de desarrollo. Uno de los más utilizados con este objetivo es UML**.** UML son las siglas de Unified Modeling Language (Lenguaje Unificado de Modelado). Es un lenguaje que permite visualizar, especificar, construir y documentar los artefactos de sistemas que involucran una gran cantidad de software, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Mediante UML es posible establecer una serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código. Es importante remarcar que UML es un "lenguaje de modelado" para especificar métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. (36)

Se decidió utilizar UML ya que el mismo se puede usar para diferentes tipos de sistemas, consolida muchas de las notaciones y conceptos más usados orientados a objetos, y es de fácil entendimiento. (37)

Una vez que se ha seleccionado el lenguaje para modelar los componentes del sistema y su desarrollo, es deseable poder contar con herramientas que permitan optimizar y hacer más fácil esta tarea. Por ello es importante seleccionar una herramienta CASE[3] para modelar que utilice

[3] Por sus siglas en inglés ***ComputerAided Software Engineering*** (Ingeniería de Software Asistida por

a UML como lenguaje.

Visual Paradigm for UML es una herramienta case de modelado visual que utiliza UML para modelar el sistema y Diagramas Entidad-Relación (DER) para el diseño de bases de datos. Cuenta con varias herramientas de diagramación que ayudan a aumentar la validez en fase de análisis y diseño.

Visual Paradigm se caracteriza por:

- Disponibilidad en múltiples plataformas.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código permanecen sincronizados en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Soporta aplicaciones Web.
- Generación de código para Java y exportación como HTML. Compatibilidad entre ediciones, entre otros. (38)

Se decidió utilizar la herramienta de modelado Visual Paradigm for UML Suite 5.0 para el desarrollo de la solución debido a la disponibilidad en múltiples plataformas que posee. No obstante solo se utilizó para la confección del Árbol de Sintaxis Abstracta (AST) de la herramienta. Además es una herramienta con licencia comercial y gratuita, facilita el desarrollo de la programación siendo capaz de generar código en varios lenguajes de programación.

Un elemento importante en los sistemas informáticos actuales lo constituyen *los Sistemas Gestores de Base de Datos (SGBD)* diseñados para gestionar grandes bloques de información, la definición de estructuras para el almacenamiento y los mecanismos para la gestión de la información. El SGBD es una aplicación que permite a los usuarios definir, crear, mantener la base de datos y proporcionar un acceso controlado. (39) Es un requisito del sistema utilizar a PostgreSQL como SGBD, es por ello que a continuación se describe el mismo.

PostgreSQL es un SGBD relacional, distribuido bajo licencia BSD[4] y con su código fuente disponible libremente. Es uno de los sistemas de gestión de bases de datos de código abierto más usado en el mercado (40). PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Es ampliamente

Computadora). Diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software.

[4] (*Berkeley Software Distribution*), se califica como una licencia mucho más libre que la GPL.

considerado como el sistema gestor de bases de datos de código abierto más avanzado del mundo; fue pionero en muchos conceptos que estuvieron disponibles en algunos sistemas de bases de datos comerciales de alto calibre como por ejemplo: control de acceso simultáneo, gestión de transacciones, puntos de seguridad; fue uno de los primeros intentos en implementar un motor de bases de datos relacional. (41)

PostgreSQL en su versión 9.2 es una petición del cliente, que está respaldada por las funcionalidades antes descritas. Además PostgreSQL posibilita alta concurrencia de usuarios accediendo de forma simultánea al sistema, siendo esta una cualidad de interés para la herramienta a elaborar. Un elemento a destacar es que en ocasiones las herramientas para el diseño de bases de datos que proporcionan los SGBD son complejas o no proveen todas las características necesarias, en estos casos se opta por herramientas proporcionadas por terceros, este es el caso de PostgreSQL, donde se decidió utilizar DBDesigner.

DBDesigner es un sistema totalmente visual para el diseño de bases de datos, que combina características y funciones profesionales con un diseño simple, muy claro y fácil de usar, a fin de ofrecer un método efectivo para gestionar bases de datos. Permite administrar la base de datos, diseñar tablas, hacer peticiones SQL manuales y mucho más. Se enfoca al desarrollo de scripts para Firebird/InterBase, Microsoft SQL Server, MySQL, Oracle o PostgreSQL. (42)

Otro de los requisitos planteados por el cliente es que el sistema sea multiplataforma, es por ello que una vez que se han decidido la metodología, la herramienta CASE y el SGBD a utilizar se debe decidir el lenguaje de programación para hacer una realidad la propuesta.

Un lenguaje de programación es un idioma artificial dentro de la computación; permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. Existen múltiples lenguajes de programación y su clasificación va desde los de bajo nivel a los de alto nivel; estos últimos permiten escribir las aplicaciones en un lenguaje más cercano al lenguaje natural. En las últimas décadas los lenguajes de programación han experimentado un desarrollo acelerado, sobre todo con el surgimiento de internet. Dentro de los lenguajes de programación más populares se pueden mencionar a C, C++, Java y C#.

Java es un lenguaje orientado a objetos, multiplataforma y se desarrolla bajo la licencia pública general de GNU (43). A continuación se mencionan algunas cualidades de Java: (44)

- **Orientación a objetos**: Java desde su concepción es un lenguaje orientado a objetos, que implementa la mayoría de los elementos de este paradigma; el elemento central

son las clases y los objetos.

- **Seguridad**: El nivel de seguridad en Java está dado por las características del lenguaje, tales como la ausencia de punteros o el ocultamiento de la información propio de la programación orientada a objetos.
- **Multiplataforma**: El funcionamiento del programa Java es el mismo en todas las plataformas y sólo cambia la apariencia, que se adapta a la del sistema operativo que lo ejecuta (windows, linux, entre otros.)
- **Arquitectura neutral:** Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos.
- **Applets[5] y desarrollo para internet**: Los applets son una manera de incluir programas complejos en el ámbito de una página web. Estos applets se programan en Java y por tanto se benefician de la potencia de este lenguaje para la red.

La versión a utilizar es Java 7.2.1, además de este lenguaje son necesarios, un entorno integrado para desarrollar la aplicación (IDE), bibliotecas que permitan el enlace con el SGBD y la construcción de la ayuda para ello se realizó también una búsqueda de información, y análisis de algunos sistemas. En el caso del entorno se optó por NetBeans. Se decidió utilizar el IDE NetBeans en su versión 7.0 para el desarrollo de la herramienta con el lenguaje de programación Java, ya que se considera idónea para el desarrollo de la solución.

NetBeans es un IDE para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Puede servir para cualquier otro lenguaje de programación. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs[6] de NetBeans, y un archivo especial que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Es un producto libre y gratuito sin restricciones de uso. (45)

En el caso del enlace con PostgreSQL se opta por la utilización de Java Database Connectivity

[5] Son programas incrustados en otras aplicaciones, normalmente una página Web que se muestra en un navegador.

[6] (*Application Programming Interface*), es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

(**JDBC**) en su versión 4. JDBC es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice. JDBC establece una conexión con una BD, envía sentencias SQL y procesa los resultados. Para usar JDBC con un sistema gestor de base de datos en particular, es necesario disponer del driver JDBC apropiado que haga de intermediario entre ésta y JDBC. Dependiendo de varios factores, este driver puede estar escrito en Java puro, o ser una mezcla de Java y métodos nativos JNI (Java Native Interface). (46)

JUnit es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java, además de ser un conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. (47)

JavaHelp es una expansión de Java que facilita la programación de las ventanas de ayuda en las aplicaciones java. Mediante esta librería en su versión 2.0.5, se pueden crear las ventanas típicas de ayuda de las aplicaciones informáticas, en las que sale en el lado izquierdo un panel con varias pestañas: índice de contenidos, búsqueda, temas favoritos, índice alfabético, entre otros. En el lado derecho sale el texto de la ayuda. Las ventanas de ayuda pueden lanzarse directamente con la pulsación de botones en la aplicación, o bien por medio de la pulsación de la tecla F1, mostrando la ayuda correspondiente a la ventana sobre la que estamos trabajando. Las ventanas de ayuda de JavaHelp se configuran por medio de varios ficheros en formato XML. Los textos de ayuda que se quieran mostrar se escribirán en ficheros con formato HTML. (48)

Conclusiones parciales

Teniendo en cuenta el apoyo que pueden brindar las TIC para mejorar el aprendizaje de los estudiantes y el estudio realizado hasta el momento a partir de la situación problémica identificada, se ha llegado a las siguientes conclusiones:

- La evaluación automatizada constituirá el principal proceso para la retroalimentación de los estudiantes en el auto-aprendizaje del CR. Esta evaluación combinada en una herramienta y puesta a disposición de los estudiantes, puede contribuir a mejorar el

desarrollo de la habilidad relacionada con la formulación de expresiones del CR.

- Durante la revisión documental no se encontraron herramientas en el ámbito nacional, y las encontradas a nivel internacional no cumplen con los requerimientos que posee la UCI.
- Una breve caracterización de las herramientas analizadas arroja que: de acuerdo al lenguaje con que trabajan el 75 % de ellas utilizan Java, la mayoría no utiliza la notación matemática habitual, algunas no realizan la traducción del CR a SQL y no poseen conexión con ningún SGBD, requisitos impuestos por el departamento que dirige el desarrollo de la aplicación.
- Para el desarrollo de la propuesta de solución se identificaron las herramientas, metodologías y lenguajes necesarios para su confección y se seleccionaron:
    - XP como metodología de desarrollo de la herramienta pues el proyecto y el equipo de desarrollo son pequeños.
    - UML como lenguaje de modelado y se utilizará Visual Paradigm for UML Suite 5.0 como herramienta de modelado.
    - El lenguaje de programación a utilizar será Java en su versión 7, y como IDE a utilizar será NetBeans 7.2.
    - Se utilizará PostgreSQL 9.2 como sistema gestor de base de datos.
    - JDBC 4 para PostgreSQL 9.2, para la comunicación con la base de datos.
    - DBDesigner Fork 1.5 es un sistema visual de diseño de bases de datos, que ofrece un método efectivo para gestionar las mismas.
    - JavaHelp en su versión 2.0.5, para la creación y edición del manual de ayuda.

Capítulo 2: Análisis y diseño de la herramienta para la evaluación automatizada de expresiones del Cálculo Relacional

El presente capítulo tiene como objetivo presentar el resultado del análisis y diseño de la propuesta de sistema. Se especifican los requisitos funcionales y no funcionales que este debe cumplir para satisfacer las necesidades del cliente, se describe la arquitectura requerida por el cliente y se definen los diferentes patrones de diseño que serán utilizados en la implementación.

2.1 Descripción del sistema

La Herramienta HEA-CRT se desarrolló como una aplicación de escritorio y se publicará en el EVEA donde estará disponible para su descarga. El sistema consta con dos aplicaciones clientes, una para la gestión de los ejercicios que será utilizada por el profesor, y otra para la solución de los ejercicios (estudiante).

El profesor será el responsable de la gestión de los ejercicios (adicionar, modificar, eliminar y listar los mismos). Como datos del ejercicio se incluye el nombre, el enunciado, el script de la BD, las relaciones de las tablas con sus atributos, los incisos con sus descripciones, la complejidad de los mismos y sus posibles soluciones en CRT y SQL. Cada nuevo ejercicio genera un esquema[7] dentro de la BD, con la información necesaria para su solución.

El estudiante deberá descargar la aplicación del EVEA, ejecutarla y realizar la conexión a la BD mediante una interfaz gráfica. Una vez realizados estos pasos, puede acceder a la interfaz “Responder Ejercicios”, seleccionar el ejercicio y el inciso, e introducir una posible solución en CRT. El estudiante oprime un botón para que sea validada por la aplicación y el sistema será el encargado de realizar validaciones léxicas, sintácticas y semánticas. Durante este proceso se ofrece una retroalimentación al usuario en caso de existir algún error.

Para la verificación de la solución, una vez que se comprueba que no existen errores de tipo léxico, sintáctico o semántico, la herramienta traduce la respuesta en CRT a SQL. Esta traducción se ejecuta en el SGBD conjuntamente con la solución en SQL propuesta por el profesor. Luego se verifica si coinciden los resultados de ambas consultas mediante la implementación de una funcionalidad que compare estos resultados, retroalimentando al

[7] Un esquema contiene tablas, vistas, procedimientos, y otros más. Se encuentra dentro de una base de datos, que a su vez está dentro de un servidor.

usuario en cualquier caso (respuesta correcta o incorrecta). Esta comparación tiene en cuenta que las filas y las columnas de las tuplas resultantes de la ejecución de ambas expresiones en SQL pueden tener un orden diferente, y sin embargo se consideran resultados iguales y luego se visualizan los datos obtenidos en cada consulta en SQL.

A continuación se muestra el flujo de información del sistema antes descrito:

Figura 2.1 Flujo de información en el sistema.

2.2 Especificación de los Requisitos del Software

Los requisitos para un sistema de software determinan lo que hará el sistema y definen las restricciones de su operación e implementación. Los mismos se clasifican en funcionales y no funcionales. Los requisitos funcionales son las capacidades o restricciones que dan vida al software, mientras que los 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.

Requisitos Funcionales

RF1: Conectar a BD (IP servidor, usuario, contraseña, nombre de la BD).

RF2: Gestionar Ejercicios.

2.1 Insertar Ejercicios (nombre, enunciado, relaciones, incisos).

2.1.1 Cargar y ejecutar script.

2.2 Modificar Ejercicios (identificador del ejercicio, enunciado, script).

2.2.1 Modificar Incisos (descripción, solución_sql, solución_crt).

2.3 Eliminar Ejercicios (identificador del ejercicio).

2.3.1 Eliminar Incisos (identificador del inciso).

2.3.2 Eliminar relaciones (identificador de la relación).

2.4 Listar Ejercicios.

RF3: Responder Ejercicios.

3.1 Seleccionar Ejercicios.

3.2 Introducir consulta en Cálculo Relacional.

RF4: Traducir solución del lenguaje Cálculo Relacional a SQL.

4.1 Validar la solución Léxicamente.

4.2 Validar la solución Sintácticamente.

4.3 Validar la solución Semánticamente.

4.4 Generar código SQL.

RF5: Comprobar Solución.

5.1 Ejecutar consulta en PostgreSQL.

5.2 Comparar resultados de las consultas del profesor y del estudiante.

5.3 Retroalimentar la solución brindada por el estudiante.

5.4 Mostrar resultados de las consultas del profesor y del estudiante.

Requisitos no funcionales

Apariencia o interfaz externa:

RnF1. El sistema contará con una interfaz amigable que permita tanto al profesor como al estudiante interactuar de forma cómoda, y que facilite el trabajo con la herramienta.

Legales:

RnF2. Se usarán herramientas de software libre y código abierto, o que funcionen bajo las licencias GNU/GPL, por lo que el sistema será desarrollado también en los términos de la licencia GNU/GPL.

Disponibilidad:

RnF3. El sistema estará disponible para su descarga y utilización en el EVEA de la Universidad, dentro del curso virtual de SBD1.

RnF4. El Servidor de BD estará disponible 24x7x365 para que los estudiantes puedan utilizar la aplicación asincrónicamente.

Confiabilidad:

RnF5. Se garantiza un tratamiento adecuado de las excepciones y la validación de las entradas del usuario.

Soporte:

RnF6. Debe poseer un manual de usuario.

Seguridad:

RnF7. Integridad: Se garantiza la integridad de la información que se maneja en el sistema ya que solo podrá ser modificada por las personas autorizadas.

Requisitos de hardware:

RnF8. Para explotación del cliente: PC Pentium 3 o superior, CPU 133 MHZ o superior, 256MB RAM mínimo, 512MB RAM recomendada o superior y interfaz de red para las conexiones con la BD.

RnF9. Para explotación del servidor: CPU Dual Core 2.0 GHZ o superior, memoria RAM de 2 GB, 10 GB o más de espacio libre en disco duro y una tarjeta de red para las conexiones.

Requisitos de software:

RnF10. El servidor deberá tener instalado el Gestor de Bases de Datos PostgreSQL 9.2.

RnF11. Tener instalado previamente en cada ordenador la Máquina Virtual de Java (JRE[8]) en la versión 7.0 o superior.

Personas y aplicaciones relacionadas con el sistema.

Persona	Descripción
Profesor	Es el encargado de gestionar los ejercicios que el estudiante va a realizar y las BD correspondientes, así como las consultas en CRT y SQL que dan solución a cada inciso.
Estudiante	Podrá ver los ejercicios publicados, seleccionar los que desee resolver, y entrar las respuestas correspondientes en el editor.
Herramienta CR	Es la encargada de realizar la validación y traducción del lenguaje CRT al SQL.

Tabla 2.1 Descripción de las personas relacionadas con el sistema.

2.5 Historias de Usuario

Las Historias de Usuario (HU) corresponden a la técnica utilizada para especificar los requisitos del software. Se trata de formatos en los cuales el cliente describe brevemente las características y funcionalidades que el sistema debe poseer, sean requisitos funcionales o no funcionales. Las HU constan de tres (3) o cuatro (4) líneas escritas por el cliente, en un lenguaje no técnico sin hacer hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados. Las HU deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Una vez realizadas estas estimaciones, se organiza una reunión de planificación, con los diversos actores del proyecto (cliente, desarrolladores, gerentes), a los efectos de establecer un plan o cronograma de entregas. (49)

[8] Java Runtime Environment, es un conjunto de utilidades que permite la ejecución de programas Java. El JRE actúa como un "intermediario" entre el sistema operativo y Java.

Listado de Historias de Usuario

Historia de Usuario	
Número: 1	Conectar a la BD
Modificación de Historia de Usuario	
Usuario: Estudiante y Profesor.	**Iteración Asignada**: 1
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: El usuario deberá conectarse a la BD antes de realizar cualquier acción en el sistema, con los siguientes parámetros: • IP del servidor. • Usuario. • Contraseña. • Nombre de la BD.	

Tabla 2.2 Descripción HU Conectar a la BD.

Historia de Usuario	
Número: 2	Gestionar ejercicio.
Modificación de Historia de Usuario	
Usuario: Profesor.	**Iteración Asignada**:1
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: Se realiza la acción de gestionar ejercicios, dentro de la cual se insertan, modifican, se eliminan y se listan los ejercicios insertados en la BD por el profesor. Dentro de la propia inserción se publicará el enunciado del ejercicio y la solución propuesta por el profesor; además, se almacenan en la BD los parámetros siguientes: nombre del ejercicio, enunciado, el script para el ejercicio, sus relaciones, incisos, las soluciones en CRT y en SQL, además de la complejidad del inciso.	

Tabla 2.3 Descripción HU Gestionar ejercicios.

Historia de Usuario	
Número: 3	Responder ejercicios.
Modificación de Historia de Usuario	
Usuario: Estudiante.	**Iteración Asignada**: 2
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: El estudiante seleccionará el ejercicio a resolver de la lista de ejercicios mostrada, luego seleccionará el inciso que desea resolver e introducirá la consulta en Cálculo Relacional de Tuplas. Una vez introducida la respuesta, presiona el botón Ejecutar.	

Tabla 2.4 Descripción HU Responder ejercicios.

Historia de Usuario	
Número:4	Traducir solución del lenguaje CRT al SQL.
Modificación de Historia de Usuario	

Usuario: Herramienta CR.	**Iteración Asignada**:3
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: Se validará la consulta entrada por el estudiante en CRT léxica, sintáctica y semánticamente. Si ocurre un error durante estas validaciones se le mostrará un mensaje al estudiante con el tipo de error cometido. Si la consulta está correcta se traducirá al lenguaje SQL.	

Tabla 2.5 Descripción HU Traducir solución del lenguaje CRT al SQL.

Historia de Usuario	
Número:5	Comprobar solución.
Modificación de Historia de Usuario	
Usuario: Herramienta AR	**Iteración Asignada**:4
Prioridad en negocio: Alta	
Riesgo en Desarrollo: Bajo	
Descripción: Una vez traducida la consulta al lenguaje SQL, se ejecuta en el SGBD PostgreSQL, y luego se ejecuta la consulta introducida por el profesor. Una vez obtenidos ambos resultados se comparan, y se realiza la retroalimentación especificándole al estudiante a través de un mensaje, si su resultado coincide con el del profesor o no. Luego se le muestra en una ventana ambos resultados para su verificación.	

Tabla 2.6 Descripción HU Comprobar solución.

2.6 Planificación

En la Planificación se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Por otra parte, el equipo de desarrollo mantiene un registro de la velocidad de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración. La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. (50)

Las **estimaciones de esfuerzo** asociada a la implementación de las HU, las establecen los programadores, para ello utilizan puntos*. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos*. (50).

Para obtener el sistema propuesto se ha realizado la estimación de esfuerzo por cada HU, como se muestra a continuación:

No. HU	Historia de Usuario	Puntos de estimación
1.	Conectar a la BD.	1
2.	Gestionar Ejercicios.	2
3.	Responder Ejercicios.	2
4.	Traducir solución del lenguaje Cálculo Relacional a SQL.	4
5.	Comprobar Solución.	2

Tabla 2.7 Estimación de esfuerzo por HU

Siguiendo la metodología XP, a continuación se **planifica la duración de las iteraciones** como parte del ciclo de vida del proyecto. Esta etapa tiene como objetivo mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de ellas. En el caso del proyecto las iteraciones se definen como:

- **Iteración 1:** Durante esta primera iteración se implementan las HU que se consideraron más necesarias atendiendo a su relevancia e impacto para la aplicación, se realizó la 1ra y 2da HU, las mismas tiene una prioridad alta dada la complejidad del sistema en su conjunto, obteniéndose así la versión 0.1 del producto dándole la posibilidad al usuario de probar dichas funcionalidades.
- **Iteración 2:** El objetivo de esta iteración es la HU 3, que se encarga de visualizar los ejercicios a resolver para su posterior solución. Una vez concluida la iteración el estudiante tendrá la posibilidad de introducir las respuestas a estos ejercicios que la aplicación ofrece.
- **Iteración 3:** Durante el transcurso de esta iteración se realizó la implementación de la HU 4, en la cual se desarrolla el analizador léxico, sintáctico y semántico, brindando la información pertinente en caso de error y la traducción del lenguaje CRT al SQL.
- **Iteración 4:** Se realiza la HU 5, donde se comparan los resultados de las consultas del profesor y del estudiante, y se brinda la retroalimentación, mostrando los resultados de ambas consultas al estudiante.

A continuación se muestra cómo quedan distribuidas las HU según el orden en que serán abordadas en cada iteración, teniendo en cuenta su prioridad y el tiempo de duración de las mismas.

Iteraciones	Orden de las Historias de Usuario a implementar en la iteración	Duración de la iteración
Iteración 1	1. Gestionar ejercicio. 2. Conectar a la BD.	3 semanas
Iteración 2	1. Responder ejercicios.	2 semanas
Iteración 3	1. Traducir solución del lenguaje CRT al SQL.	4 semanas
Iteración 4	1. Comprobar solución.	2 semanas

Tabla 2.8 Planificación de la duración de las iteraciones.

La **planificación de entregas** establece qué HU serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todas las personas del proyecto (cliente, desarrolladores, gerentes, entre otros.) (49). Estas entregas se realizan en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores.

A entregar	Final 1ra iteración	Final 2da iteración	Final 3ra iteración	Final 4ta iteración
Herramienta para la evaluación automatizada de expresiones del lenguaje AR, herramienta para la evaluación automatizada de expresiones del lenguaje CRT.	30/3/2013 Versión 0.1	13/4/2013 Versión 0.2	11/5/2013 Versión 0.3	25/5/2013 Versión 1.0

Tabla 2.9 Planificación de entregas de las iteraciones.

2.7 Diseño

El diseño es, durante el ciclo de vida del software, la etapa donde se aplican distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización física. Con el diseño se pretende construir un sistema que se ajuste a las limitaciones impuestas y que cumpla con los requisitos de la aplicación, logrando que los cambios o modificaciones futuras puedan realizarse de manera más sencilla. XP no define una técnica específica de modelado, pueden utilizarse indistintamente esquemas sencillos, tarjetas CRC (Clase, Responsabilidad, Colaboración) o diagramas de clase utilizando UML, siempre que sean útiles y no requieran mucho tiempo en su creación. (35) En el presente epígrafe se exponen los elementos relevantes para el diseño del sistema.

La propuesta de solución está basada en la **arquitectura Cliente/Servidor**, esta arquitectura

consiste básicamente en un cliente que realiza peticiones a otro programa (el servidor) que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora, es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras. La interacción cliente-servidor es el soporte de la mayor parte de la comunicación por redes. Ayuda a comprender las bases sobre las que están construidos los algoritmos distribuidos. (51)

Figura 2.2 Arquitectura Cliente – Servidor

Conjuntamente con la selección de la arquitectura Cliente/Servidor se determinó el uso del estilo arquitectónico en tres capas para las aplicaciones clientes, con el objetivo de estructurar la implementación para soportar requerimientos complejos operacionales y garantizar mantenibilidad, reusabilidad, robustez y seguridad. Garlan y Shaw definen el estilo en capas como una organización jerárquica, de manera que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior. La arquitectura en tres capas propuesta para la construcción de las aplicaciones clientes, está conformada por las capas: Presentación, Negocio y Acceso a datos. (52)

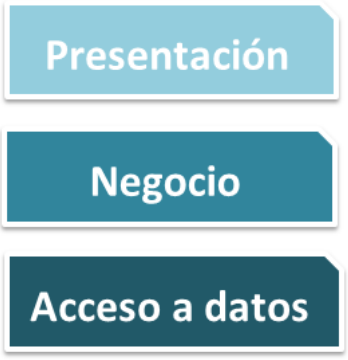

Figura 2.3 Arquitectura en tres capas.

La capa de Presentación contiene las interfaces con que el usuario (estudiante o profesor) interactúa. Esta se comunica con la capa de Negocio, que contiene la lógica de implementación fundamental de la herramienta: el traductor CRT y las clases encargadas de la gestión de

ejercicios. La capa de Acceso a datos es la encargada de establecer las conexiones con la BD mediante la clase Conexión.

Patrones de diseño.

Un patrón de diseño constituye una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haberse comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias. (53)

Patrones GOF

Existen diversos patrones de diseño, dentro de ellos se encuentran los denominados Patrones GOF. Estos patrones facilitan la localización de los objetos que formarán el sistema, especifican interfaces para las clases, especifican implementaciones, facilitan el aprendizaje y la comunicación entre programadores y diseñadores. Estos patrones se clasifican en:

- Creacionales: Resuelven problemas relativos a la creación de objetos.
- Estructurales: Resuelven problemas relativos a la composición de objetos.
- De comportamiento: Resuelven problemas relativos a la interacción entre objetos.

Dentro de los patrones Creacionales podemos encontrar varios tipos como: Factory Method (método de fabricación), Prototype (prototipo), Singleton (instancia única), dentro de las Estructurales se encuentran: Composite (Objeto compuesto), Decorator (Decorador), Facade (Fachada), entre otros; y los patrones de Comportamiento son: Chain of Responsibility (Cadena de responsabilidad), Interpreter (Intérprete), Visitor (Visitante), y algunos más. A continuación se explicarán los patrones usados en la confección de la herramienta: (54)

- **Composite** (Objeto compuesto): Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, y permitir a los clientes tratar objetos simples y compuestos de modo uniforme. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera. Se puede observar en el AST de la Herramienta. (Ver Figura 3.1)
- **Chain of Responsibility** (Cadena de responsabilidad): Es un patrón de comportamiento que proporciona a más de un objeto la capacidad de atender una petición, para así evitar el acoplamiento con el que objeto que hace la petición. Se forma con estos objetos una cadena, en la cual cada objeto satisface la petición o la pasa al siguiente. (55) Se puede ver a la hora de manejar las responsabilidades en las clases del análisis léxico, sintáctico y semántico.

- **Interpreter** (Intérprete): Dado un lenguaje, define una representación para su gramática junto con un intérprete que usa dicha representación para interpretar sentencias en ese lenguaje. (55) Este patrón se pone de manifiesto en la realización de la gramática definida para el lenguaje CRT. (Ver Tabla 3.1)
- **Visitor** (Visitante): Representa una operación que está pensada para ser aplicada sobre los elementos de una estructura de objetos, en el caso del sistema en la utilización del AST, permitiendo así definir y añadir un nuevo comportamiento sin necesidad de cambiar las clases de los elementos de la estructura de objetos. (55)

Modelo Entidad Relación

El Modelo Entidad Relación (MER) es uno de los varios modelos conceptuales existentes para el diseño de bases de datos. Fue inventado por Peter Chen en los años setenta. El propósito de este modelo es simplificar el diseño de bases de datos a partir de descripciones textuales de los requerimientos. (56)

Los elementos esenciales del modelo son las *entidades*, los *atributos* y las *relaciones* entre las entidades. Una *entidad* es un objeto que existe y que es distinguible de otros objetos. Una entidad puede ser concreta (ejemplo un profesor) o abstracta (como un algoritmo, un curso o una dirección en Internet). Las entidades tienen atributos. Un *atributo* de una entidad es una característica interesante sobre ella, es decir, representa alguna propiedad que interesa almacenar. Por ejemplo, el profesor tiene un nombre, una fecha de nacimiento, entre otros datos. Una *relación* es una asociación entre entidades, generalmente dos. Una relación puede ocurrir entre dos entidades de un mismo conjunto de entidades (por ejemplo, un empleado es supervisado por su jefe, quien a su vez es otro empleado), o, más corrientemente, entre entidades de conjuntos distintos (por ejemplo, un curso es dictado por un profesor). Pueden existir relaciones entre más de dos conjuntos de entidades. (56)

La figura 2.4 muestra el modelo relacional de la BD inicial de la HEA-CRT, que se le irán adicionando tantos esquemas como ejercicios se inserten, de manera que en cada esquema se incluyan las tablas relacionadas con cada ejercicio.

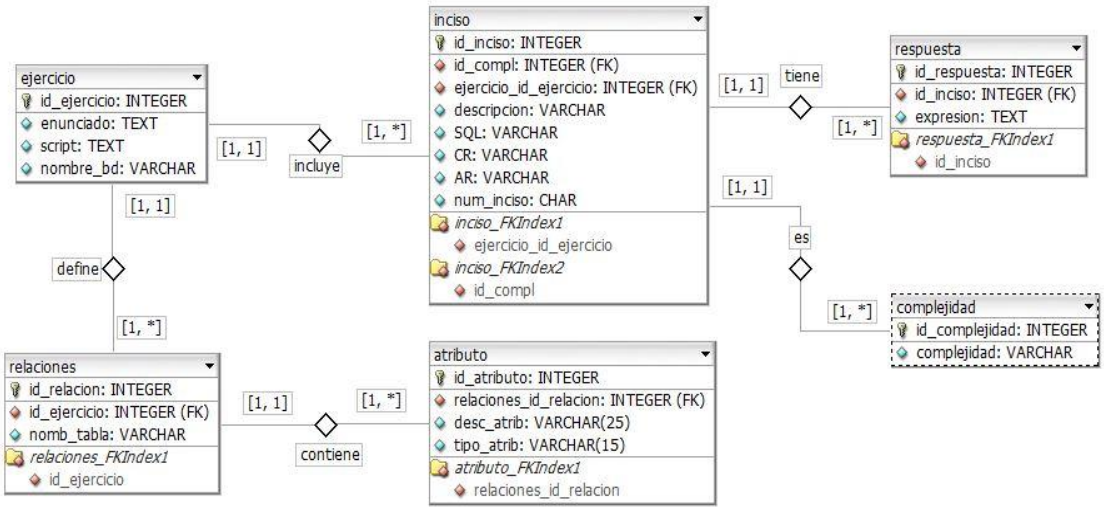

Figura 2.4 Diseño de la BD de la herramienta.

Tarjetas CRC (Clase-Responsabilidad-Colaboración).

El objetivo de las tarjetas CRC es realizar un inventario de las clases necesarias para implementar el sistema y la forma en que van a interactuar, para con ello facilitar el análisis y discusión de las mismas por parte de los integrantes del equipo de proyecto. Esto propicia que el diseño sea lo más simple posible, verificando las especificaciones del sistema. El uso de estas tarjetas permite al programador centrarse y apreciar el desarrollo orientado a objetos, olvidándose de los malos hábitos de la programación procedural clásica. Las tarjetas CRC permiten trabajar con una metodología orientada a objetos, y cada tarjeta representa una clase. (57)

Clase Conexión	
Descripción: clase encargada de realizar la conexión a la BD.	
Responsabilidad	**Colaborador**
Establecer conexión con la BD.	Conexión.
Cerrar conexión con la BD.	
Verificar estado de la conexión.	

Tabla 2.10 Tarjeta CRC Conexión.

Clase Traductor	
Descripción: clase encargada de validar y traducir las consultas.	
Responsabilidad	**Colaborador**
Validar y traducir las consultas introducidas en el sistema.	Traductor.

Tabla 2.11 Tarjeta CRC Traductor.

Clase Interfaz del Traductor	
Descripción: clase encargada de visualizar los datos.	
Responsabilidad	**Colaborador**
Se encargará de obtener los datos y enviarlos a la clase Traductor.	Traductor. Conexión. Interfaz del Traductor.

Tabla 2.12 Tarjeta CRC Interfaz del Traductor.

Prototipos de interfaz de usuario.

Un prototipo es un modelo (representación, demostración o simulación) fácilmente ampliable y modificable de un sistema planificado, probablemente incluyendo su interfaz y su funcionalidad de entradas y salidas. (58) El prototipo de interfaz de usuario (IU) permite comprobar cuestiones tales como la navegación, visualización de los atributos definidos, así como la ejecución de todos los servicios. A continuación se muestran algunos de los prototipos de la aplicación: (Ver Anexo 3)

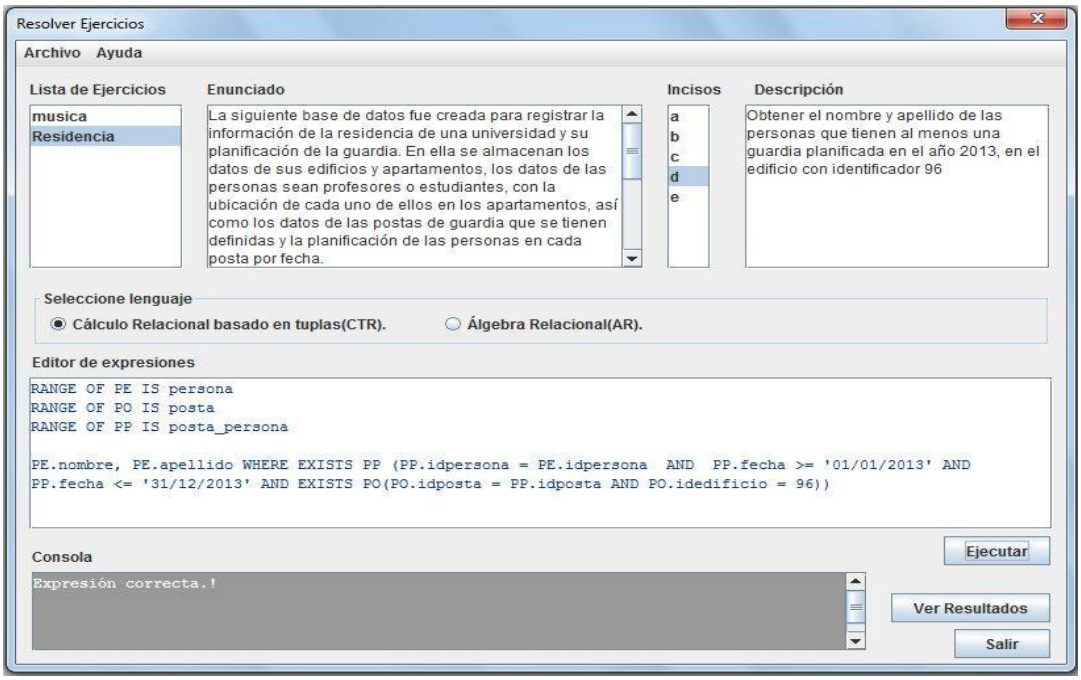

Figura 2.5 Interfaz Resolver Ejercicios.

Conclusiones parciales

En el presente capítulo se abordaron temas relacionados con el análisis y el diseño de la propuesta de solución que se desea implementar.

- Mediante la descripción elaborada del sistema, el cliente será capaz de expresar, comprender y aprobar el funcionamiento de la aplicación, que aunque no es una aplicación compleja en su interacción sí lo es en su construcción.

- Los requisitos funcionales y no funcionales en conjunto con las HU y la descripción de las mismas, ayudaron a definir las condiciones y capacidades que deben estar presentes en el sistema para satisfacer las necesidades del cliente. Las cinco (5) HU definidas permitirán a los programadores desarrollar su trabajo y conocer la estimación del tiempo para el desarrollo de la herramienta.
- A partir de las HU a implementar se definieron cuatro (4) iteraciones; se definió además la duración de cada iteración y el orden en la implementación de las HU. Con la planificación de entregas se definió una fecha límite aproximada de setenta y siete (77) días, para la culminación del producto final.
- La arquitectura a utilizar es una arquitectura Cliente/Servidor, ya que esta se corresponde con las necesidades definidas para la herramienta. Se definieron los patrones GOF a utilizar para alcanza un desarrollo organizado y un producto de software con calidad.

Capítulo 3: Implementación y pruebas de la HEA-CRT

En este capítulo se abordará lo referente a la implementación y pruebas de la HEA-CRT. En un producto guiado por la metodología XP el intercambio con el cliente es fundamental, además del uso de estándares que sean entendibles por todo el equipo, y la programación dirigida por las pruebas unitarias realizadas por los desarrolladores. La definición de estas pruebas condiciona o “dirige” el desarrollo de la herramienta y la programación en parejas, ambos trabajando juntos en un mismo ordenador, lo cual posee las siguientes ventajas:

- La mayoría de los errores se descubren en el momento en que se codifican, ya que el código es permanentemente revisado por dos personas.
- La cantidad de defectos encontrados en las pruebas es estadísticamente menor.
- Los diseños son mejores, el código más óptimo y el equipo resuelve problemas en forma más rápida.
- Las personas aprenden más acerca del sistema.

Las personas aprenden a trabajar juntas, generando mejor dinámica de grupo y haciendo que la información fluya rápidamente. Y por último la propiedad colectiva del código, que todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, cualquier pareja de programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o recodificar, independientemente de llevar un ritmo sostenido en todo el proceso de implementación. (35)

A partir de lo definido en el capítulo anterior es necesario desarrollar las dos aplicaciones con que cuenta la herramienta, la que le permitirá al profesor diseñar los ejercicios y la que permitirá que el estudiante resuelva y verifique las soluciones a los ejercicios planteados por el profesor.

3.1 Tareas de Ingeniería

Para desarrollar una correcta implementación de las HU descritas por el cliente se deben definir por parte del equipo de desarrollo las Tareas de Ingeniería que se realizarán en cada una de las iteraciones. Las TI, también conocidas como tareas de implementación, permiten a los

desarrolladores obtener un nivel de detalle más avanzado que el que propician las HU. (59) El desarrollo del software se planificó en cuatro iteraciones de trabajo. A continuación se muestran algunas HU correspondientes a la primera iteración de las TI: (Ver Anexo 5)

TI para la iteración #1:

HU #1: Conectar a la BD.

Tarea de Ingeniería		
Número de Tarea:1	Número de la HU:1	
Nombre de Tarea: Conectar a la BD.		
Tipo de Tarea: Desarrollo	Fecha Inicio: 9/3/2013	Fecha Fin: 11/3/2013
Programadores Responsables: Damaysis Carmona Hernández Lidiagnis Fonseca Pérez		
Descripción: El sistema muestra la opción para que el usuario (profesor o estudiante) inserte los ejercicios en el sistema. Además se crea la BD (IP servidor, usuario, contraseña, nombre de la BD) y se ejecuta el script.		

Tabla 3.1 Tarea de Ingeniería Conectar a la BD.

HU #1: Gestionar ejercicios.

Tarea de Ingeniería		
Número de Tarea: 2	Número de la HU:1	
Nombre de Tarea: Insertar ejercicios		
Tipo de Tarea: Desarrollo	Fecha Inicio: 11/3/2013	Fecha Fin: 15/3/2013
Programadores Responsables: Alexei Pupo Ricardo Adnier Castellanos Puentes.		
Descripción: El sistema muestra la opción para que el usuario (profesor) inserte los ejercicios en el sistema y se ejecute el script.		

Tabla 3.2 Tarea de Ingeniería Insertar ejercicios.

Tarea de Ingeniería		
Número de Tarea: 3	Número de la HU:1	
Nombre de Tarea: Modificar ejercicios		
Tipo de Tarea: Desarrollo	Fecha Inicio: 16/3/2013	Fecha Fin: 22/3/2013
Programadores Responsables: Damaysis Carmona Hernández Lidiagnis Fonseca Pérez		
Descripción: El sistema muestra la opción para que el usuario (profesor) modifique los ejercicios en el sistema. Esto incluye modificación de los ejercicios en la BD.		

Tabla 3.3 Tarea de Ingeniería Modificar ejercicios.

Tarea de Ingeniería		
Número de Tarea: 4	Número de la HU:1	
Nombre de Tarea: Eliminar ejercicios		
Tipo de Tarea: Desarrollo	Fecha Inicio: 23/3/2013	Fecha Fin: 30/3/2013
Programadores Responsables: Damaysis Carmona Hernández		

Lidiagnis Fonseca Pérez
Descripción: El sistema muestra la opción para que el usuario (profesor) elimine los ejercicios en el sistema. Esto incluye la eliminación de los ejercicios en la BD.

Tabla 3.4 Tarea de Ingeniería Eliminar ejercicios

3.2 Implementación y validación del traductor de expresiones del Cálculo Relacional a expresiones SELECT del SQL.

Uno de los elementos fundamentales del proyecto es la conversión de expresiones del CR a expresiones SELECT del SQL, para poder validar la expresión formulada por los estudiantes. En el epígrafe 1.4 se estudiaron los elementos fundamentales de un traductor entre dos lenguajes. Este epígrafe se le dedica al proceso de implementación y validación del traductor.

3.2.1 Gramática del Cálculo Relacional

El traductor recibe como entrada un programa escrito en un lenguaje fuente (expresiones de CR), del cual se necesita conocer cómo se escribe y cuáles son las palabras y símbolos que se utilizan. Para la definición de las palabras y símbolos se utilizan expresiones regulares y para describir el lenguaje se utilizan las gramáticas. Para definir el sentido de las sentencias en el lenguaje se utiliza una descripción asociada a las reglas gramaticales.

Una gramática es un conjunto de reglas para formar correctamente las frases o expresiones, y constituye la mejor vía para la descripción sintáctica de los lenguajes, (27) por lo cual se identificaron los elementos necesarios del lenguaje CR para la elaboración de la misma.

Para describir el lenguaje de las expresiones válidas en el CR se definió la gramática **GCR** = {**N**, **Σ**, **P**, **S**}, donde **N** es el conjunto de los símbolos no terminales o alfabeto de variables del lenguaje, **Σ** el conjunto de símbolos terminales o alfabeto de constantes, **P** el conjunto de reglas de producción y **S** el símbolo inicial de la gramática, por donde comienza la generación de sentencias.

Esta gramática se obtuvo luego de convertir la gramática original presentada en los documentos de la asignatura a una gramática LL(1), la que permite implementar un analizador sintáctico descendente y predictivo. A continuación se presentan los elementos que definen a la gramática.

N = {<Programa_Cálculo>, <Declaración_Tuplas>, <Más_Tuplas>, <Tuplas>,
<Más_Relación>, <Declaración_Relación>, <Relación>, <Término>,
<Expresión_Cálculo>,<Var_Cualificadas>,<Cualificada><Más_Cualificada>,
<Predicado>, <Más_Condición>, <Comp_Oper>, <Valores>,<Booleanos>,
<Más_Valores>, <Aritmética>, <Otra_Aritmética>, <Sexo>, <Comparar>,
<Más_Comparar>, <Cuantificador>, <Tipo_Dato>, <Exp_Aritmética>,

<Más_Término>, <Más_Factor>, <Decl_Atributo>,<Factor>,
<Op_Comparación>, <Op_logicos>, <Lista_de_atributos>, <Más_Atributos>}

Σ = { SEPARADORES, (,), :,’,’, ;, +,-, *, /, =, <, >, <>, <=, >=, RELATION, RANGE, OF, IS, WHERE, AND, OR, NOT, EXISTS, FORALL,‘, _, “, TRUE, FALSE, M, F, IDENTIFICADORES}

S = <Programa_Cálculo>

P:

<Programa_Cálculo>	→	<Declaración_Relación><Declaración_Tuplas><Expresión_Cálculo>
<Declaración_Tuplas>	→	<Tuplas><Más_Tuplas>
<Más_Tuplas>	→	**,** <Declaración_Tuplas>|**e**
<Tuplas>	→	**RANGE OF Identificador IS Identificador**
<Declaración_Relación>	→	<Relación><Más_Relación>
<Más_Relación>	→	**,** <Declaración_Relación>|**e**
<Relación>	→	**RELATION Identificador IS** (<Lista_de_atributos>) **;**
<Expresión_ Cálculo>	→	<Var_Cualificadas> **WHERE** <Predicado>
<Var_Cualificadas>	→	<Cualificada><Más_Cualificada>
<Más_Cualificada>	→	**,** <Var_Cualificadas> | **e**
<Cualificada>	→	**Identificador.Identificador**
<Predicado>	→	<Comp_Oper><Más_Condición>
<Más_Condición>	→	<Op_logicos> <Predicado><Mas_Condicion>| e
<Comp_Oper>	→	**NOT** <Cuantificador> **(**<Valores>**)** | < Cuantificador > **(**<Valores>**)** | <Valores>
<Valores>	→	<Cualificada> <Mas_Valores>
<Más_Valores>	→	<Op_Comparación> <Comparar> | **=** <Más_Comparar>
<Aritmética>	→	<Exp_aritmética> <Op_Comparación > <Otra_Aritmética>
<Otra_Aritmética>	→	<Exp_Aritmética > | <Aritmética> |e
<Comparar>	→	<Hora> |<Fecha>| <Exp_Aritmetica>
<Más_Comparar>	→	<Booleanos> | **Literal_Cadena** | <Sexo> | <Exp_Aritmetica>
< Cuantificador >	→	**EXISTS Identificador** | **FORALL Identificador**
<Exp_Aritmética>	→	<Término><Más_Término>
<Más_Término>	→	**+** <Término><Más_Término>| **-** <Término><Más_Término>
<Término>	→	<Factor><Más_Factor>
<Más_Factor>	→	***<Factor><Más_Factor>| **/**<Factor><Más_Factor>
<Factor>	→	(<Aritmética >)| **literal_entero**| **literal_real| <**Cualificada **>**
<Op_Comparación>	→	=|<|>|<=|>=|<>
<Op_logicos>	→	**AND** | **OR**
<Lista_de_atributos>	→	**<**Decl_Atributo**>**<Más_Atributos>
<Más_Atributos>	→	**,** <Lista_de_atributos> | **e**
< Decl_Atributo>	→	**Identificador**: <Tipo_Dato>
<Tipo_Dato>	→	**Int**|**float**|**String**|**Boolean**|**Char**|**Date**|**Time**
<Sexo>	→	**M** | **F**
<Booleanos>	→	**TRUE** | **FALSE**

Tabla 3.5 Gramática de la HEA-CRT.

3.2.2 Analizador léxico

En la fase del análisis léxico se almacenan en la tabla de símbolos los *tokens* reconocidos por el *scanner,* recibidos como cadenas de caracteres del programa fuente. Para implementar el analizador léxico se definieron las expresiones regulares para cada tipo de elemento del lenguaje, operadores, literales e identificadores.

A continuación se muestran algunas de las expresiones regulares más significativas:

letra	→	a|…|z|A|…|Z |
dígito	→	1|2|3|4|5|6|7|8|9
entero	→	(dígito)(dígito |0)*|0
parte_decimal	→	(dígito|0)*(dígito)|0
real	→	entero.parte_decimal
hora	→	‘dígito dígito : dígito dígito : dígito dígito’
fecha	→	‘dígito dígito /dígito dígito / dígito dígito dígito dígito’
cadena	→	“(letra |’ ‘| _ | dígito)*”
identificador	→	letra (letra | _ |dígito)*(letra | dígito)

Luego de definir las expresiones regulares correspondientes a los tokens del lenguaje, se construyó para cada una de ellas un autómata finito determinista que permitiera reconocer las cadenas asociadas a los lenguajes generados por las expresiones regulares. Después de obtenidos estos autómatas, los mismos se integraron en uno, elaborando un diagrama de transición, y se pasó a la implementación el método *Proximo_Token* de la clase *Léxico*.

A continuación se muestra un autómata y su correspondiente código:

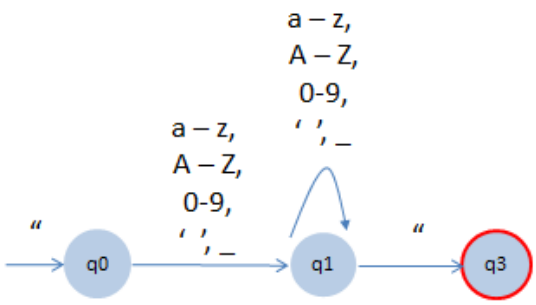

Figura 3.1 Autómata literal de cadena.

```
private Token Literal_Cadena() throws IOException{
    StringBuilder buf = new StringBuilder();
    do {
        buf.append(c_actual);
        Consumir();
        if (c_actual == '\0') {
            throw new IOException("Se esperaba \"\" .");
        }
    } while (esDigito() || esLetra() || c_actual == ' ' || c_actual != '"');
    Consumir();
    return new Token(Tipo_token.Cadena, buf.toString(), entrada.getLinea_actual(),
                    tabla_simbolos.Adicionar(buf.toString(), Tipo_token.Cadena));
}
```

Figura 3.2 Implementación del autómata literal de cadena.

Seguidamente se muestra un ejemplo de una expresión de CRT y los tokens reconocidos por el analizador léxico:

PR.categoria WHERE PR.idedificio = 95

Token	Lexema
tk_identificador	“PR”
tk_punto	“.”
tk_identificador	“categoria”
tk_WHERE	“WHERE”
tk_identificador	“AP”
Tk_punto	“.”
tk_identificador	“idedificio”
tk_igualQue	“=”
tk_literalEntero	“95”

Tabla 3.6 Expresión en Cálculo Relacional y tokens reconocidos.

3.2.3 Analizador sintáctico

En la etapa del analizador sintáctico se reciben los tokens generados por el *scanner* y se verifica si la sintaxis en la secuencia de tokens se corresponde con la estructura sintáctica del lenguaje CR. Para la correcta detección de la sintaxis del lenguaje se implementó un analizador sintáctico descendente recursivo predictivo, con la correspondiente gramática del lenguaje CR de tipo LL(1).

La correcta validación del analizador sintáctico suministra a la fase posterior (Análisis Semántico) el Árbol de Sintaxis Abstracta el cual se construye a partir de la secuencia de tokens recibida y utilizando la jerarquía de clases definida. Para devolver el correspondiente AST se construyó una jerarquía de clases que pudiera representar los elementos que

componen la sintaxis del CR con sus correspondientes relaciones. Una sección de la jerarquía de clases, la relacionada con los predicados, se muestra a continuación:

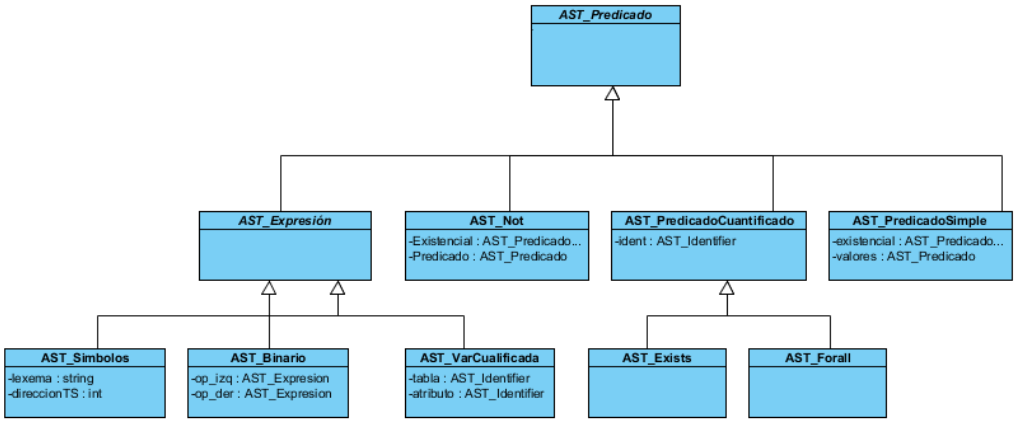

Figura 3.3 Jerarquía del AST Predicado y sus descendientes.

La clase Sintáctico (*Parser*) cuenta con 29 métodos relacionados con los no terminales de la gramática definida. Estos métodos son los que permiten verificar que la sintaxis de la secuencia de tokens se corresponde con la definida en la gramática del lenguaje. A continuación se muestra un ejemplo de la regla y su correspondiente implementación en el analizador sintáctico.

<Programa_Cálculo> -> <Declaración_Relación> <Declaración_Tuplas> <Expresión_Cálculo>

```
public AST_ProgramaC ProgramaCalculoR() throws IOException {

    //declaracion de variables
    List<AST_Dec_Relacion> dec_relacion;
    List<AST_DeclTupla> dec_tuplas;
    AST_Expresion_Calc exp_calculo;

    dec_relacion = new LinkedList<>();
    dec_relacion = Declaracion_Relaciones(dec_relacion);

    dec_tuplas = new LinkedList<>();
    dec_tuplas = Declaracion_Tuplas(dec_tuplas);


    exp_calculo = Expresion_Calculo();

    //retornar la expresion de calculo
    return new AST_ProgramaC(dec_relacion,dec_tuplas,
                             exp_calculo,actual.getLinea());
}
```

Figura 3.4 Implementación de la regla Programa Cálculo.

<Expresión_Cálculo> -> <Var_Cualificadas> WHERE <Predicado>

```
//<Expresión_Cálculo >→<Var_Cualificadas> WHERE <Predicado>
private AST_Expresion_Calc Expresion_Calculo()throws IOException{

    //declaracion de variables
    int linea = actual.getLinea();
    List<AST_VarCualificada> list_variables;

    list_variables = new LinkedList<>();
    list_variables = Var_Cualificadas(list_variables);

    Match(Tipo_token.WHERE);
    AST_Predicado predicado = Predicado();
    if (actual.getTipo() != Tipo_token.FDE ) {
        reportes_error.add(new Error_sintactico(actual.getLinea(),
                "Ha escrito una expresión incorrecta. Se obtuvo: "
                +actual.getLexema()));
    }

    return new AST_Expresion_Calc(list_variables,predicado,linea);
}
```

Figura 3.5 Implementación de la regla Expresión_Cálculo del Analizador Sintáctico.

<Cualificada> -> *ident_tabla.ident_atributo*

```
//<Cualificada>→tabla.atributo
private AST_VarCualificada Cualificada()throws IOException{
    AST_Identifier ident = null;
    AST_Identifier identAtributo = null;

    if (actual.getTipo() == Tipo_token.Identificador)
    {
       ident = new AST_IdentifierRef(actual.getLexema(),
               actual.getEntrada(), actual.getLinea());
    }
    Match(Tipo_token.Identificador);
    Match(Tipo_token.Punto);
    if (actual.getTipo() == Tipo_token.Identificador)
    {
       identAtributo = new AST_IdentifierRef(actual.getLexema(),
               actual.getEntrada(), actual.getLinea());
    }
    Match(Tipo_token.Identificador);
    return new AST_VarCualificada(ident, identAtributo, actual.getLinea());
}
```

Figura 3.6 Implementación de la regla Cualificada del Analizador Sintáctico.

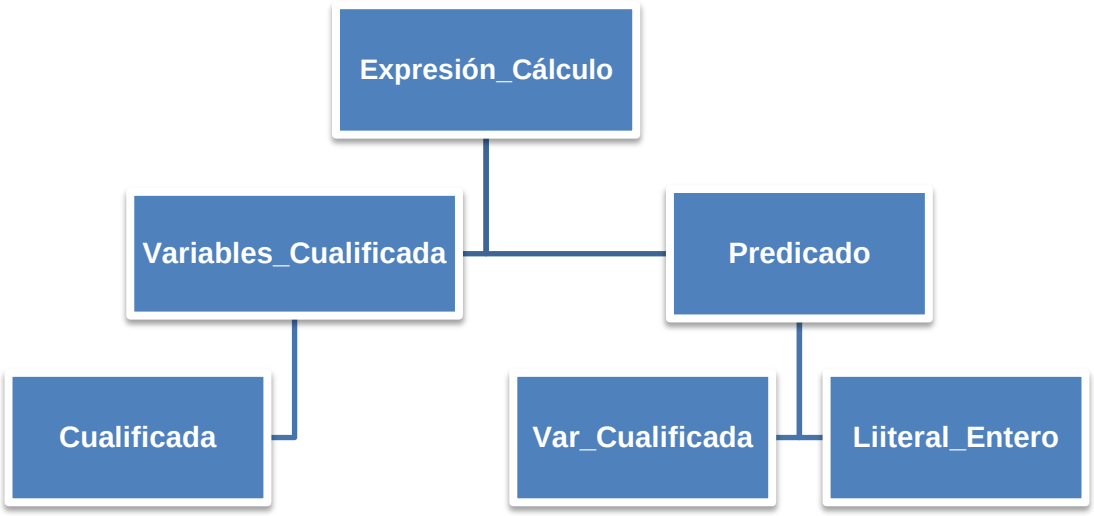

Figura 3.7 AST generado por el análisis sintáctico.

3.2.4 Analizador semántico

La etapa de análisis semántico del traductor toma como entrada el AST generado por el analizador sintáctico una vez validada correctamente la sintaxis del lenguaje CR y realiza reglas semánticas para validar dicho árbol. De las validaciones más importantes realizadas sobre el AST se encuentra las comprobaciones de tipos y la correspondencia entre los atributos utilizados y declarados en las relaciones. A continuación se muestran tres métodos donde se puede observar la implementación de estas validaciones.

```
public Object Visitar(AST_Suma o) {
    Tipos_Datos tipo1 = (Tipos_Datos) o.getIzq().Visitar(this);
    Tipos_Datos tipo2 = (Tipos_Datos) o.getDer().Visitar(this);
    if (tipo1 != tipo2) {
        if ((tipo1 == Tipos_Datos.Float && tipo2 == Tipos_Datos.Integer) ||
              (tipo1 == Tipos_Datos.Integer && tipo2 == Tipos_Datos.Float))
        {
            return Tipos_Datos.Float;
        }
        reportes.add(new Error_semantico(o.getLinea(),
                "Incompatibilidad de datos en la operación '+': "
                + tipo1.toString() + " y " + tipo2.toString()));
        return Tipos_Datos.NoDefinido;

    } else {
        return tipo1;
    }
}
```

Figura 3.8 Implementación del Visitor AST_Suma.

```
    public Object Visitar(AST_IdentifierRef o) {
        Info_Simbolo info = ts.Obtener_nodo(o.getDireccion_TS());
        if (!info.isDeclarado()) {
            reportes.add(new Error_semantico(o.getLinea(),
                    "Referencia a un Identificador no declarado:" + o.getLexema()));
        }
        return info.getTipodato();
    }
```

Figura 3.9 Implementación del Visitor AST_IndentifierRef.

```
public Object Visitar(AST_VarCualificada o) {
    Info_Simbolo tabla = ts.Obtener_nodo(o.getTabla().getDireccion_TS());
    if (!tabla.isDeclarado() || !tabla.isTabla()) {
        reportes.add(new Error_semantico(o.getTabla().getLinea(),
                "Referencia a un Identificador de tabla no declarado: "
                + o.getTabla().getLexema()));
    }
    Info_Simbolo atrib = ts.Obtener_nodo(o.getAtributo().getDireccion_TS());
    if (!atrib.isDeclarado() || !atrib.isAtributo()) {
        reportes.add(new Error_semantico(o.getAtributo().getLinea(),
                "Referencia a un Identificador de atributo no declarado: "
                + o.getAtributo().getLexema()));
        return Tipos_Datos.NoDefinido;
    }
    return atrib.getTipodato();
}
```

Figura 3.10 Implementaciónn del Visitor AST_VarCualificada.

Como resultado de esta fase de obtiene un AST decorado con la información adicional obtenida durante el análisis semántico. En esta etapa además se completa la información de la tabla de símbolos.

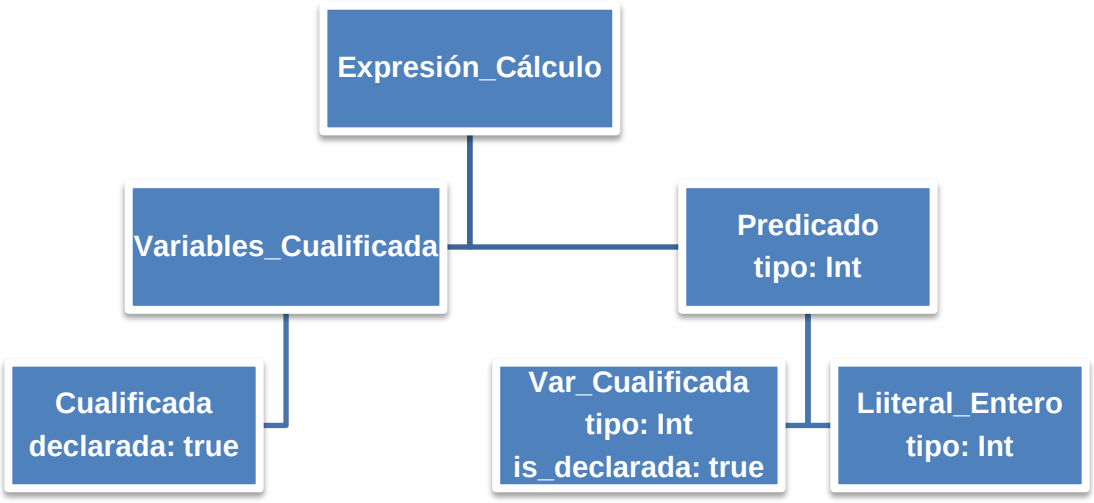

Figura 3.11 AST decorado.

3.2.5 Generador de código

Para la etapa de generación de código se definieron reglas de equivalencia del lenguaje CR al SQL, logrando transformar las expresiones de CR a expresiones de consulta de SQL. Una de las equivalencias que resultó interesante es la transformación del cuantificador universal FORALL a una expresión que utilice el cuantificador existencial EXIST. A continuación se muestran los fragmentos de código que permiten generar la parte correspondiente a los cuantificadores dentro de la expresión en SQL. Esta implementación se realizó dentro de la generación asociada a un predicado simple.

FORALL	NOT EXISTS(SELECT * FROM tabla WHERE NOT(p))

```
/*
 * equivalencia en SQL del EXISTS
 */
String identExists = o.getExistencial().Generador(this);
int p  = tablas_tuplas.indexOf(identExists);
existencial= true;
return "EXISTS( SELECT * FROM "+tablas_tuplas.get(p)
        +" WHERE " +o.getPredicado().Generador(this)+")";
```

Figura 3.12 Implementación del cuantificador EXISTS en SQL.

```
/*
 * equivalencia en SQL del FORALL
 */
String identExists = o.getExistencial().Generador(this);
int p  = tablas_tuplas.indexOf(identExists);
existencial= true;
return "NOT EXISTS( SELECT * FROM "+tablas_tuplas.get(p)
        +" WHERE NOT (" +o.getPredicado().Generador(this)+"))";}
```

Figura 3.13 Implementación del cuantificador FORALL en SQL.

```
@Override
public String Generador(AST_Not o) {

    if (o.getExistencial() instanceof AST_Forall) {
        /*
         * equivalencia del NOT FORALL al SQL
         */
        String identExists = o.getExistencial().Generador(this);
        int p  = tablas_tuplas.indexOf(identExists);
         existencial= true;
        return "EXISTS( SELECT * FROM "+tablas_tuplas.get(p)+
                " WHERE NOT(" +o.getPredicado().Generador(this)+"))";
    }
    else{
        /*
         * Equivalencia del NOT EXISTS al SQL
         */
        String identExists = o.getExistencial().Generador(this);
        int p  = tablas_tuplas.indexOf(identExists);
         existencial= true;
        return "NOT EXISTS( SELECT * FROM "+tablas_tuplas.get(p)+
                " WHERE NOT(" +o.getPredicado().Generador(this)+"))";
    }
}
```

Figura 3.14 Implementación del NOT EXISTS, NOT FORALL al SQL.

Uno de los elementos interesantes en esta transformación lo constituye la utilización de subconsultas y la expresión SELECT del SQL.

A continuación se muestra un ejemplo de la transformación de una expresión del CR correspondiente al modelo que se incluye en el Anexo 4 a su expresión SQL correspondiente.

*ED.idedificio **WHERE EXISTS** AP (AP.idedificio = ED.idedificio **AND***
***EXISTS** PE (PE.idedificio = AP.idedificio **AND** PE.idapto = AP.idapto))*

```
SELECT edificio.idedificio
FROM edificio
WHERE EXISTS (
 SELECT * FROM apartamento WHERE apartamento.idedificio = edificio.idedificio AND
 EXISTS( SELECT * FROM persona WHERE persona.idedificio = apartamento.idedificio
 AND persona.idapto = apartamento.idapto) )
```

El proceso descrito garantizó obtener una clase que a partir de una expresión del CR permite transformarla en una expresión SQL equivalente. Esta clase puede ser integrada a diferentes aplicaciones lo que constituye su principal potencialidad. Una vez que se obtuvo esta clase se definieron las tareas de ingeniería para la implementación de las HU definidas en el capítulo anterior.

3.3 Pruebas Unitarias

Son pruebas dirigidas a probar clases aisladamente, están relacionadas con el código y la responsabilidad de cada clase y sus fragmentos de código más críticos. Las pruebas unitarias se realizan para asegurar la calidad del código entregado. Es la mejor forma de detectar errores tempranamente en el desarrollo, ayuda a definir los requerimientos y responsabilidades de cada método en cada clase probada, permite incluso hacer pruebas de estrés tempranamente en el código. (60) Las pruebas unitarias se fueron desarrollando a medida que se terminaba de implementar alguna funcionalidad, para lo cual se utilizó el framework JUnit. (Ver Anexo 6)

3.4 Pruebas de aceptación

En esta prueba se evalúa el grado de calidad del software con relación a todos los aspectos relevantes para que el uso del producto se justifique. Se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema soporte. (61) Estas pruebas se realizaron mediante el tipo de prueba funcional, utilizando el método Caja Negra y específicamente la técnica de Partición Equivalente. (33)

A las HU pertenecientes a la iteración 1 se desarrollaron 3 iteraciones de pruebas y como resultado se detectaron dos no conformidades para la 1ra iteración de prueba, una en la segunda y en la 3ra iteración fueron mitigadas todas.

A la HU perteneciente a la iteración 2 se desarrollaron 2 iteraciones de pruebas y como resultado se detectó una no conformidad para la 1ra iteración y en la 2da iteración fue mitigada.

A la HU perteneciente a la iteración 3 se desarrollaron 3 iteraciones de pruebas y como resultado se detectaron dos no conformidades para la 1ra iteración, dos en la segunda y en la 3ra iteración fueron mitigadas todas.

A la HU perteneciente a la iteración 4 se desarrollaron 2 iteraciones de pruebas y como resultado se detectó una no conformidad para la 1ra iteración, siendo mitigada en la 2da iteración.

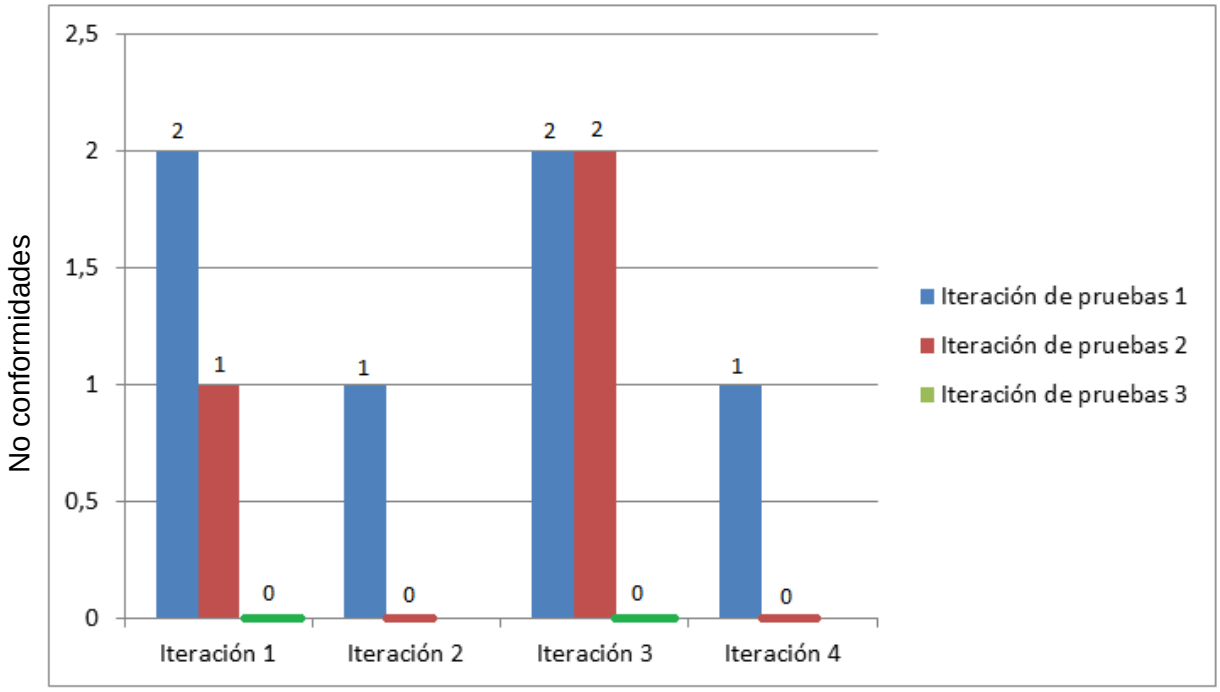

Figura 3.15 Resultados de las iteraciones de prueba.

No conformidades

No.	No conformidad	Ubicación	Estado
1	No se muestra un mensaje indicando que se ha realizado la conexión con la BD.	Prueba de Aceptación Iteración #1, HU Conectar a la BD.	Resuelta
2	No se elimina el ejercicio seleccionado de la BD.	Prueba de Aceptación Iteración #1, HU Gestionar Ejercicios\ Eliminar Ejercicios.	Resuelta
3	No se muestra un mensaje indicando que no existen ejercicios en la BD.	Prueba de Aceptación Iteración #1, HU Gestionar Ejercicios\ Listar Ejercicios.	Resuelta
4	No se muestra un mensaje indicando que no seleccionó el inciso a resolver.	Prueba de Aceptación Iteración #2, HU Responder Ejercicios.	Resuelta
5	No se muestran los errores Sintácticos	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Cálculo	Resuelta

No.	No conformidad	Ubicación	Estado
		Relacional de Tuplas a SQL	
6	No se muestran los errores semánticos.	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Cálculo Relacional de Tuplas a SQL	Resuelta
7	No traduce correctamente la expresión del cuantificador EXISTS.	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Cálculo Relacional de Tuplas a SQL	Resuelta
8	No traduce correctamente la expresión del cuantificador FORALL.	Prueba de Aceptación Iteración # 3, HU Traducir solución del lenguaje Cálculo Relacional de Tuplas a SQL	Resuelta
9	No se muestra un mensaje especificando que los resultados de las consultas no coinciden	Prueba de Aceptación Iteración # 4, HU Comprobar Solución.	Resuelta

Tabla 3.7 No conformidades del sistema.

A continuación se observan algunas de las Pruebas de aceptación realizadas a la HEA-CRT, las demás se encuentran en el Anexo 7.

Conectar a la BD.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario (estudiante o profesor) accede a la Interfaz Principal y selecciona la opción "Conectarse a la BD" e introduce los datos: -IP -Usuario -Contraseña -Nombre de la BD Luego presiona el botón Conectar. -10.51.17.146 -user -****** -Ejercicios		El sistema se conecta a la BD y muestra un mensaje verificando la conexión.	El sistema se conecta a la BD y muestra un mensaje verificando la conexión.	Se deben introducir todos los datos y estar correctos.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
	El usuario accede a la Interfaz Principal y selecciona la opción "Conectarse a la BD" e introduce los datos: - -user -****** -Ejercicios	El sistema muestra un mensaje especificando que se debe entrar el IP.	El sistema muestra un mensaje especificando que se debe entrar el IP.	El IP debe estar compuesto por números y tener el formato IPv4.
	El usuario accede a la Interfaz Principal y selecciona la opción "Conectarse a la BD" e introduce los datos: -10.51.17.146 - -****** -Ejercicios	El sistema muestra un mensaje especificando que se debe entrar el usuario.	El sistema muestra un mensaje especificando que se debe entrar el usuario.	
	El usuario accede a la Interfaz Principal y selecciona la opción "Conectarse a la BD" e introduce los datos: -10.51.17.146 -user - -Ejercicios.	El sistema muestra un mensaje especificando que se debe entrar la contraseña.	El sistema muestra un mensaje especificando que se debe entrar la contraseña.	
	El usuario accede a la Interfaz Principal y selecciona la opción "Conectarse a la BD" e introduce los datos: -10.51.17.146 -user	El sistema muestra un mensaje especificando que se debe entrar el nombre de la BD.	El sistema muestra un mensaje especificando que se debe entrar el nombre de la BD.	

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
	-****** -			

Tabla 3.8 Prueba de Aceptación – Conectar a la BD.

Responder Ejercicios.

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
El usuario estudiante accede a la interfaz Responder Ejercicios y selecciona el ejercicio a responder. Luego selecciona el inciso e introduce la consulta en Cálculo Relacional de ese inciso. Seguidamente presiona el botón Ejecutar.		El sistema verifica que la entrada es válida y se introducen los datos en la aplicación.	El sistema verifica que la entrada es válida y se introducen los datos en la aplicación.	
	El usuario estudiante accede a la interfaz Responder Ejercicios y no selecciona el ejercicio, Luego presiona el botón Ejecutar.	El sistema muestra un mensaje indicando que debe seleccionar un ejercicio para su solución.	El sistema muestra un mensaje indicando que debe seleccionar un ejercicio para su solución.	
	El usuario estudiante accede a la interfaz Responder Ejercicios y	El sistema muestra un cartel indicando que se debe seleccionar un inciso.	El sistema muestra un cartel indicando que se debe seleccionar un inciso.	

Clases Válidas	Clases Inválidas	Resultado Esperado	Resultado de la Prueba	Observaciones
	selecciona el ejercicio, y presiona el botón Ejecutar.			
	El usuario estudiante accede a la interfaz Responder Ejercicios, selecciona el ejercicio y el inciso, y presiona el botón Ejecutar.	El sistema muestra un cartel indicando que se debe introducir la consulta en Cálculo Relacional.	El sistema muestra un cartel indicando que se debe introducir la consulta en Cálculo Relacional	

Tabla 3.9 Prueba de Aceptación – Responder ejercicios.

Conclusiones parciales

En el presente capítulo se muestran los resultados de la implementación y pruebas realizadas a la herramienta, se ha llegado a las siguientes conclusiones:

- A partir de la notación definida por el Departamento Metodológico Central de Ingeniería y Gestión de Software para la enseñanza y aprendizaje del lenguaje CRT en la Universidad, se logró definir una gramática para este lenguaje.
- Se implementó un traductor entre el lenguaje del CR y el SQL, para ello fue necesario implementar un analizador léxico, un analizador sintáctico, un analizador semántico y un generador de código para la traducción. El mismo puede ser integrado a otros proyectos que requieran de la misma traducción.
- Se realizaron las tareas de ingeniería correspondientes a cada HU para obtener un nivel más detallado de la implementación de la herramienta.
- Se realizaron pruebas unitarias a la implementación del código durante todo el desarrollo de la herramienta, lo que permitió comprobar en todo momento la validez de la misma.
- Se realizaron pruebas de aceptación que permitieron detectar tres no conformidades en la primera iteración, una en la segunda iteración, cuatro en la tercera iteración y una en la cuarta iteración, las cuales fueron resueltas satisfactoriamente.

Conclusiones

Una vez terminada la investigación se puede concluir que:

- A partir de la revisión documental se pudo comprobar el papel que posee la utilización de las TIC en la educación, así como las ventajas que proporciona, en especial en lo relacionado con la evaluación. Se asume que la evaluación automatizada es aquella que se realiza a través de un sistema informático.
- El análisis realizado de las herramientas encontradas en el ámbito nacional e internacional para la evaluación automatizada de expresiones del CRT arrojó que aunque existen algunas herramientas que son utilizadas en la enseñanza del CRT, estas no cumplen con las necesidades planteadas por el Departamento para la UCI.
- A partir de las especificaciones dadas por el departamento y del análisis del proceso de desarrollo de software se seleccionaron: XP como metodología de desarrollo de software, Visual Paradigm for UML y DBDesigner y como herramientas de modelado, PostgreSQL como SGBD, Java como lenguaje de programación, JDBC como librería para la comunicación con la BD, NetBeans como IDE, JUnit para la realización de las pruebas unitarias y JavaHelp para la creación y edición de la ayuda.
- Al aplicar la metodología XP, fue necesario elaborar 5 historias de usuario para describir el sistema en el lenguaje del cliente; realizar la estimación de esfuerzo, que arrojó un tiempo de desarrollo de 11 semanas y la planificación de 4 entregas. El sistema utiliza una arquitectura Cliente-Servidor y para las aplicaciones clientes la arquitectura en tres capas y en su implementación se utilizaron los Patrones de Diseño GOF.
- Se realizó la implementación de un traductor del lenguaje CRT al lenguaje SQL. Este proceso se realizó a partir de la obtención de las expresiones regulares de cada elemento del lenguaje y de la gramática facilitada por el departamento.
- La herramienta elaborada cuenta con dos aplicaciones, una de ellas permite que el profesor diseñe los ejercicios y la otra permite al estudiante resolverlos, retroalimentándolo con los errores cometidos y/o con los resultados de la consulta. Ambas son aplicaciones de escritorio y cuenta cada una con una interfaz gráfica.
- Durante la fase de prueba, se realizaron pruebas de aceptación para validar las HU definidas y pruebas unitarias para validar la implementación del código, con las cuales se logró corregir y/o validar correctamente el estado de la herramienta.

- Como principal resultado se logró desarrollar una herramienta que permite a los estudiantes autoevaluar su aprendizaje del contenido CRT en la asignatura SBD1, sin depender del profesor, lo que facilita la retroalimentación y su independencia, al no tener que esperar que el profesor revise los ejercicios.

Recomendaciones

En el cumplimiento de los objetivos propuestos durante el desarrollo del presente trabajo, surgieron diversas propuestas que serían recomendables a tener en cuenta para un futuro perfeccionamiento del software:

- Integrar la herramienta al entorno virtual de aprendizaje para hacer uso de las utilidades que brinda esta plataforma para la gestión de usuarios, gestión de reportes, gestión de calificaciones, entre otras.
- Realizar la validación de incisos cuyo conjunto de resultado es vacío y en los cuales no se puede utilizar la estrategia de comparar los conjuntos de datos de la consulta construida a partir de la expresión del cálculo y de la solución proporcionada por el profesor.
- Realizar una implementación para validar los resultados a partir de la equivalencia entre dos AST que representen a la misma solución, omitiendo por ejemplo los nombres de la tuplas.
- Brindar recomendaciones o pistas a los estudiantes sobre los elementos que hacen que sus respuestas no sean correctas para fortalecer la retroalimentación.
- Incluir en la herramienta del profesor la posibilidad de realizar análisis sobre los resultados de un determinado ejercicio.
- Emplear el uso de técnicas de Inteligencia Artificial para validar la similitud entre las posibles soluciones a ejercicios.

Referencias Bibliográficas

1. José Ramón Gómez Pérez. *Las TIC en Educación.* [En línea] 2007. [Citado el: 4 de noviembre de 2012.] http://boj.pntic.mec.es/jgomez46/ticedu.htm..

2. Las TIC, aliadas de la Revolución cubana. *La pupila insomne.* [En línea] 22 de junio de 2012. [Citado el: 14 de febrero de 2013.] http://lapupilainsomne.wordpress.com/2012/06/22/las-tic-aliadas-de-la-revolucion-cubana/.

3. Los desafíos de las TIC para el cambio educativo. *Publicaciones.* [En línea] 2009. [Citado el: 7 de noviembre de 2012.] http://www.oei.es/publicaciones/detalle_publicacion.php?id=10. 2..

4. Historia. *Universidad de las Ciencias Informáticas.* [En línea] 2012. [Citado el: 18 de noviembre de 2012.] http://www.uci.cu/historia.

5. *Entorno Virtual de Aprendizaje.* [En línea] 2013. [Citado el: 23 de enero de 2013.] http://eva.uci.cu/.

6. Pregrado. *Universidad de las Ciencias Informáticas.* [En línea] 2012. [Citado el: 12 de diciembre de 2012.] http://www.uci.cu/pregrado.

7. Formación. *Intranet.* [En línea] 2013. [Citado el: 18 de diciembre de 2012.] http://intranet2.uci.cu/node/86/ingenieria-software.

8. *Introducción a los Sistemas de Base de Datos.* Date, C. J. México : PEARSON EDUCACIÓN, 2001. 968-444-419-2.

9. Sistemas de Bases de Datos I. . *Entorno Virtual de Aprendizaje.* [En línea] 2013. [Citado el: 15 de noviembre de 2012.] http://eva.uci.cu/course/view.php?id=180.

10. León, Lic. Yamilka Gómez. MODELO DE PLANIFICACIÓN Y CONTROL DEL PROCESO DOCENTE (P1). *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2012. [Citado el: 15 de diciembre de 2012.] http://eva.uci.cu/file.php/180/1._Documentos_Generales/Programa_y_P1/P1_SBD1_Curso_2012-2013.pdf.

11. *Introducción a los Sistemas de Bases de Datos.* México : s.n., 2001. ISBN: 968-444-419-2.

12. González, Francisco Ruiz. *Cálculo Relacional.* Castilla- La Mancha : s.n.

13. La importancia de las TIC en la educación superior. *Aula Virtual.* [En línea] noviembre de 2011. [Citado el: 23 de mayo de 2013.] http://aula.virtual.ucv.cl/wordpress/importancia-tic-en-la-edu-sup/.

14. Cueva Carrión Samanta Patricia, Pacheco Montoya Emma Patricia,Rodriguez Morales Germania del Rocio,Santos Delgado Ana Alexandra. *Técnologías de Información y Comunicación (TIC´s) en la Educación Superior.* 2009.

15. LCMS y objetos de aprendizaje. *Revista UNAM.* [En línea] 10 de noviembre de 2004. [Citado el: 23 de mayo de 2013.] http://www.revista.unam.mx/vol.5/num10/art66/int66.htm. 1607 - 6079.

16. Mateo, J. A. *La evaluación educativa, su práctica y sus metáforas.* Barcelona : s.n., 2000.

17. Conde, Mª José Rodríguez. Aplicación de las TIC a la evaluación de alumnos universitarios. [En línea] [Citado el: 19 de diciembre de 2012.] http://campus.usal.es/~teoriaeducacion/rev_numero_06_2/n6_02_art_rodriguez_conde.htm.

18. Albert Gras Martí, Marisa Cano Villalba,M. Pardo Casado,A. Celdrán Mallol,J.V. Santos Benito,J.A. Miralles Torres,M.J. Caturla Terol. *La evaluación, como ejemplo de integración de las TIC en la enseñanza.* 2003.

19. Soler, J. *Entorno virtual para el aprendizaje y evaluación automática de bases de datos.* Girona, España : s.n., 2010. Tesis doctoral. ISBN: 978-84-694-0260-3.

20. Lianni Yadira Figueredo Jiménez, Misael Rodríguez Urrutia. *"Simulador para la asignatura Sistemas Operativos".* Ciudad de la Habana : s.n., 2010. Tesis.

21. Carlos Matos Carbonell, Arnolis Rodríguez del Valle. *Creación de un sistema virtual de Transmisión de Datos para el apoyo a la docencia en la asignatura de T eleinformática I.* Ciudad de la Habana : s.n., 2008.

22. Roxana Angela Basulto López, Lianet Guevara Hernández. *Sistema de gestión del expediente docente en la asignatura de Práctica Profesional.* Ciudad de La Habana : s.n., 2009.

23. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2013. [Citado el: 2013 de abril de 18.] http://eva.uci.cu/mod/resource/view.php?id=26564.

24. CÁLCULO RELACIONAL. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2010. [Citado el: 2 de diciembre de 2012.] http://eva.uci.cu/file.php/180/2._Clases/Semana_8/1ra_frecuencia/MApoyo/Calculo_Relacional.pdf.

25. *Universidad Central de Colombia Sede Manizales.* [En línea] [Citado el: 16 de enero de 2013.] http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap4-9.html.

26. Cálculo. *Cálculo Relacional.* [En línea] 2010. [Citado el: 14 de diciembre de 2012.] http://calculo-deogracias2010.blogspot.com/2010_04_01_archive.html.

27. Martínez, Msc. Karel Osorio Ramírez y Dr. Liesner Acevedo. Técnicas de Compilación: Manual Práctico para estudiantes de Informática. v1.4.4. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2011. [Citado el: 17 de diciembre de 2012.] http://eva.uci.cu/mod/resource/view.php?id=6624.

28. *Ingeniería Informática.* 2010-2011.

29. Introducción al uso de WinRDBI. [En línea] [Citado el: 17 de diciembre de 2012.] http://www.eas.asu.edu/~cse412/winrdbi.html.

30. Lic. YamilkaGómezLeón, Dr.C Edistio VerdeciaMartínez, Ms.C.AilecGrandaDihigo. *ESTADO ACTUAL DE LAS HERRAMIENTAS DE APOYO A LA ENSEÑANZA DEL ÁLGEBRA Y CÁLCULO RELACIONAL.* La Habana : s.n., 2013.

31. Letelier, Juan Pablo. *Proceso de desarrollo de software .* Valencia : s.n., 2003.

32. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2004.

33. S. Pressman, Roger. *Software Engineering. A Practitioner's Approach.* New York : s.n., 2006. ISBN 978-0-07-337597-7.

34. José H. Canós, Patricio Letelier y M.Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n. ISSN 46022.

35. Beck, Kent. *Extreme Programming Explained.* 1999 : s.n. ISBN:0201616416.

36. UML. [En línea] 11 de noviembre de 2012. [Citado el: 11 de enero de 2013.] http://www.uml.org/.

37. Lozano, Daniel. Ventajas y Desventajas de UML . *UML.* [En línea] 16 de abril de 2009. [Citado el: 5 de febrero de 2013.] http://dlozanouml40089.blogspot.com/2009/04/ventajas-y-desventajas-de-uml.html.

38. Visual Paradigm for UML. [En línea] 18 de junio de 2012. [Citado el: 9 de enero de 2013.] http://www.visual-paradigm.com/product/vpuml.

39. Alvarez, Sara. Sistemas gestores de bases de datos. *Desarrollo Web.* [En línea] 31 de julio de 2007. [Citado el: 3 de abril de 2013.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

40. Grupo de Generación de Rankings de Classora. Ranking de las mejores bases de datos actuales. *Classora.* [En línea] 18 de enero de 2011. [Citado el: 9 de mayo de 2013.] http://es.classora.com/reports/x46901/ranking-de-las-mejores-bases-de-datos-actuales.

41. Sobre PostgreSQL. [En línea] 10 de febrero de 2012. [Citado el: 12 de enero de 2013.] http://www.postgresql.org.es/sobre_postgresql.

42. DBDesigner. [En línea] 2013. [Citado el: 11 de abril de 2013.]

http://dbdesigner.softonic.com/.

43. Conozca más sobre la tecnología Java. *JAVA.* [En línea] [Citado el: 9 de enero de 2013.] http://www.java.com/es/about/.

44. Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas. *Fesabid 98.* [En línea] [Citado el: 28 de marzo de 2013.] http://www.ciepi.org/fesabid98/Comunicaciones/m_enjolras.htm.

45. NetBeans . [En línea] 2012. [Citado el: 9 de enero de 2013.] http://netbeans.org/index_es.html.

46. Domingo, José. Manual de Java (JDBC). [En línea] [Citado el: 4 de abril de 2013.] www.josedomingo.org/web/pluginfile.php/254/mod_resource/content/0/jdbc/manual_JDBC.pdf.

47. Primeros Pasos con JUnit. *Programación .* [En línea] 2013. [Citado el: 9 de mayo de 2013.] http://www.programacion.com/articulo/primeros_pasos_con_junit_265.

48. JavaHelp System. *Java.net.* [En línea] 3 de octubre de 2007. [Citado el: 15 de mayo de 2013.] https://javahelp.java.net/.

49. Castillo Oswaldo, Figueroa Daniel, Sevilla Hector. Fases de la Programación Extrema. *Programación Extrema.* [En línea] [Citado el: 27 de febrero de 2013.] http://programacionextrema.tripod.com/fases.htm.

50. Ciclo de vida de un proyecto XP. *sourceforge.net.* [En línea] septiembre de 2005. [Citado el: 25 de febrero de 2013.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

51. Arquitectura Cliente Servidor. *Desarrollo Web.* [En línea] 30 de agosto de 2007. [Citado el: 17 de marzo de 2013.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

52. Carlos Reynoso, Nicolás Kiccillof. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

53. Patrones de diseño. *kioskea.net.* [En línea] 2013. [Citado el: 26 de marzo de 2013.] http://es.kioskea.net/contents/genie-logiciel/design-patterns.php3.

54. Prieto, Félix. *Patrones de diseño.* Valladolid, España : s.n., 2008/2009.

55. *Patrones del "Gang of Four".* Madrid, España. : s.n.

56. Universidad Central de Colombia. *Universidad Nacional de Colombia.* [En línea] [Citado el: 18 de abril de 2013.] http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap2-1.html.

57. Desarrollo de software. Tarjetas CRC. *Jummp.* [En línea] 10 de enero de 2012. [Citado el: 27 de marzo de 2013.] http://jummp.wordpress.com/2012/01/10/desarrollo-de-software-tarjetas-crc/.

58. Prototipos. *scribd.* [En línea] 2013. [Citado el: 9 de mayo de 2013.] http://es.scribd.com/doc/29597239/DEFINICION-DE-PROTOTIPO.

59. Priol, Ing. Sebastian. *Programación Extrema.*

60. Jorge Rodriguez. *Pruebas unitarias.* 2006.

61. Gestión de Calidad y Pruebas de Software. *Pruebas de Software.* [En línea] 2005. [Citado el: 3 de mayo de 2013.] http://pruebasdesoftware.com/pruebadeaceptacion.htm.

Bibliografía

1. José Ramón Gómez Pérez. *Las TIC en Educación.* [En línea] 2007. [Citado el: 4 de noviembre de 2012.] http://boj.pntic.mec.es/jgomez46/ticedu.htm..

2. Las TIC, aliadas de la Revolución cubana. *La pupila insomne.* [En línea] 22 de junio de 2012. [Citado el: 14 de febrero de 2013.] http://lapupilainsomne.wordpress.com/2012/06/22/las-tic-aliadas-de-la-revolucion-cubana/.

3. Los desafíos de las TIC para el cambio educativo. *Publicaciones.* [En línea] 2009. [Citado el: 7 de noviembre de 2012.] http://www.oei.es/publicaciones/detalle_publicacion.php?id=10. 2..

4. Historia. *Universidad de las Ciencias Informáticas.* [En línea] 2012. [Citado el: 18 de noviembre de 2012.] http://www.uci.cu/historia.

5. *Entorno Virtual de Aprendizaje.* [En línea] 2013. [Citado el: 23 de enero de 2013.] http://eva.uci.cu/.

6. Pregrado. *Universidad de las Ciencias Informáticas.* [En línea] 2012. [Citado el: 12 de diciembre de 2012.] http://www.uci.cu/pregrado.

7. Formación. *Intranet.* [En línea] 2013. [Citado el: 18 de diciembre de 2012.] http://intranet2.uci.cu/node/86/ingenieria-software.

8. *Introducción a los Sistemas de Base de Datos.* Date, C. J. México : PEARSON EDUCACIÓN, 2001. 968-444-419-2.

9. Sistemas de Bases de Datos I. . *Entorno Virtual de Aprendizaje.* [En línea] 2013. [Citado el: 15 de noviembre de 2012.] http://eva.uci.cu/course/view.php?id=180.

10. León, Lic. Yamilka Gómez. MODELO DE PLANIFICACIÓN Y CONTROL DEL PROCESO DOCENTE (P1). *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2012. [Citado el: 15 de diciembre de 2012.] http://eva.uci.cu/file.php/180/1._Documentos_Generales/Programa_y_P1/P1_SBD1_Curso_2012-2013.pdf.

11. *Introducción a los Sistemas de Bases de Datos.* México : s.n., 2001. ISBN: 968-444-419-2.

12. González, Francisco Ruiz. *Cálculo Relacional.* Castilla- La Mancha : s.n.

13. La importancia de las TIC en la educación superior. *Aula Virtual.* [En línea] noviembre de 2011. [Citado el: 23 de mayo de 2013.] http://aula.virtual.ucv.cl/wordpress/importancia-tic-en-la-edu-sup/.

14. Cueva Carrión Samanta Patricia, Pacheco Montoya Emma Patricia,Rodriguez Morales Germania del Rocio,Santos Delgado Ana Alexandra. *Técnologías de Información y Comunicación (TIC´s) en la Educación Superior.* 2009.

15. LCMS y objetos de aprendizaje. *Revista UNAM.* [En línea] 10 de noviembre de 2004. [Citado el: 23 de mayo de 2013.] http://www.revista.unam.mx/vol.5/num10/art66/int66.htm. 1607 - 6079.

16. Mateo, J. A. *La evaluación educativa, su práctica y sus metáforas.* Barcelona : s.n., 2000.

17. Conde, Mª José Rodríguez. Aplicación de las TIC a la evaluación de alumnos universitarios. [En línea] [Citado el: 19 de diciembre de 2012.] http://campus.usal.es/~teoriaeducacion/rev_numero_06_2/n6_02_art_rodriguez_conde.htm.

18. Albert Gras Martí, Marisa Cano Villalba,M. Pardo Casado,A. Celdrán Mallol,J.V. Santos Benito,J.A. Miralles Torres,M.J. Caturla Terol. *La evaluación, como ejemplo de integración de las TIC en la enseñanza.* 2003.

19. Soler, J. *Entorno virtual para el aprendizaje y evaluación automática de bases de datos.* Girona, España : s.n., 2010. Tesis doctoral. ISBN: 978-84-694-0260-3.

20. Lianni Yadira Figueredo Jiménez, Misael Rodríguez Urrutia. *"Simulador para la asignatura Sistemas Operativos".* Ciudad de la Habana : s.n., 2010. Tesis.

21. Carlos Matos Carbonell, Arnolis Rodríguez del Valle. *Creación de un sistema virtual de Transmisión de Datos para el apoyo a la docencia en la asignatura de T eleinformática I.* Ciudad de la Habana : s.n., 2008.

22. Roxana Angela Basulto López, Lianet Guevara Hernández. *Sistema de gestión del expediente docente en la asignatura de Práctica Profesional.* Ciudad de La Habana : s.n., 2009.

23. INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2013. [Citado el: 2013 de abril de 18.] http://eva.uci.cu/mod/resource/view.php?id=26564.

24. CÁLCULO RELACIONAL. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2010. [Citado el: 2 de diciembre de 2012.] http://eva.uci.cu/file.php/180/2._Clases/Semana_8/1ra_frecuencia/MApoyo/Calculo_Relacional.pdf.

25. *Universidad Central de Colombia Sede Manizales.* [En línea] [Citado el: 16 de enero de 2013.] http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap4-9.html.

26. Cálculo. *Cálculo Relacional.* [En línea] 2010. [Citado el: 14 de diciembre de 2012.] http://calculo-deogracias2010.blogspot.com/2010_04_01_archive.html.

27. Martínez, Msc. Karel Osorio Ramírez y Dr. Liesner Acevedo. Técnicas de Compilación: Manual Práctico para estudiantes de Informática. v1.4.4. *Entorno Virtual de Enseñanza y Aprendizaje.* [En línea] 2011. [Citado el: 17 de diciembre de 2012.] http://eva.uci.cu/mod/resource/view.php?id=6624.

28. *Ingeniería Informática.* 2010-2011.

29. Introducción al uso de WinRDBI. [En línea] [Citado el: 17 de diciembre de 2012.] http://www.eas.asu.edu/~cse412/winrdbi.html.

30. Lic. YamilkaGómezLeón, Dr.C Edistio VerdeciaMartínez, Ms.C.AilecGrandaDihigo. *ESTADO ACTUAL DE LAS HERRAMIENTAS DE APOYO A LA ENSEÑANZA DEL ÁLGEBRA Y CÁLCULO RELACIONAL.* La Habana : s.n., 2013.

31. Letelier, Juan Pablo. *Proceso de desarrollo de software .* Valencia : s.n., 2003.

32. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. *El proceso unificado de desarrollo de software.* Madrid : Addison Wesley, 2004.

33. S. Pressman, Roger. *Software Engineering. A Practitioner's Approach.* New York : s.n., 2006. ISBN 978-0-07-337597-7.

34. José H. Canós, Patricio Letelier y M.Carmen Penadés. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n. ISSN 46022.

35. Beck, Kent. *Extreme Programming Explained.* 1999 : s.n. ISBN:0201616416.

36. UML. [En línea] 11 de noviembre de 2012. [Citado el: 11 de enero de 2013.] http://www.uml.org/.

37. Lozano, Daniel. Ventajas y Desventajas de UML . *UML.* [En línea] 16 de abril de 2009. [Citado el: 5 de febrero de 2013.] http://dlozanouml40089.blogspot.com/2009/04/ventajas-y-desventajas-de-uml.html.

38. Visual Paradigm for UML. [En línea] 18 de junio de 2012. [Citado el: 9 de enero de 2013.] http://www.visual-paradigm.com/product/vpuml.

39. Alvarez, Sara. Sistemas gestores de bases de datos. *Desarrollo Web.* [En línea] 31 de julio de 2007. [Citado el: 3 de abril de 2013.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

40. Grupo de Generación de Rankings de Classora. Ranking de las mejores bases de datos actuales. *Classora.* [En línea] 18 de enero de 2011. [Citado el: 9 de mayo de 2013.] http://es.classora.com/reports/x46901/ranking-de-las-mejores-bases-de-datos-actuales.

41. Sobre PostgreSQL. [En línea] 10 de febrero de 2012. [Citado el: 12 de enero de 2013.] http://www.postgresql.org.es/sobre_postgresql.

42. DBDesigner. [En línea] 2013. [Citado el: 11 de abril de 2013.]

http://dbdesigner.softonic.com/.

43. Conozca más sobre la tecnología Java. *JAVA.* [En línea] [Citado el: 9 de enero de 2013.] http://www.java.com/es/about/.

44. Beneficios del uso de JAVA en las aplicaciones modernas de Bibliotecas. *Fesabid 98.* [En línea] [Citado el: 28 de marzo de 2013.] http://www.ciepi.org/fesabid98/Comunicaciones/m_enjolras.htm.

45. NetBeans . [En línea] 2012. [Citado el: 9 de enero de 2013.] http://netbeans.org/index_es.html.

46. Domingo, José. Manual de Java (JDBC). [En línea] [Citado el: 4 de abril de 2013.] www.josedomingo.org/web/pluginfile.php/254/mod_resource/content/0/jdbc/manual_JDBC.pdf.

47. Primeros Pasos con JUnit. *Programación .* [En línea] 2013. [Citado el: 9 de mayo de 2013.] http://www.programacion.com/articulo/primeros_pasos_con_junit_265.

48. JavaHelp System. *Java.net.* [En línea] 3 de octubre de 2007. [Citado el: 15 de mayo de 2013.] https://javahelp.java.net/.

49. Castillo Oswaldo, Figueroa Daniel, Sevilla Hector. Fases de la Programación Extrema. *Programación Extrema.* [En línea] [Citado el: 27 de febrero de 2013.] http://programacionextrema.tripod.com/fases.htm.

50. Ciclo de vida de un proyecto XP. *sourceforge.net.* [En línea] septiembre de 2005. [Citado el: 25 de febrero de 2013.] http://oness.sourceforge.net/proyecto/html/ch05s02.html.

51. Arquitectura Cliente Servidor. *Desarrollo Web.* [En línea] 30 de agosto de 2007. [Citado el: 17 de marzo de 2013.] http://www.desarrolloweb.com/articulos/arquitectura-cliente-servidor.html.

52. Carlos Reynoso, Nicolás Kiccillof. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

53. Patrones de diseño. *kioskea.net.* [En línea] 2013. [Citado el: 26 de marzo de 2013.] http://es.kioskea.net/contents/genie-logiciel/design-patterns.php3.

54. Prieto, Félix. *Patrones de diseño.* Valladolid, España : s.n., 2008/2009.

55. *Patrones del "Gang of Four".* Madrid, España. : s.n.

56. Universidad Central de Colombia. *Universidad Nacional de Colombia.* [En línea] [Citado el: 18 de abril de 2013.] http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060029/lecciones/cap2-1.html.

57. Desarrollo de software. Tarjetas CRC. *Jummp.* [En línea] 10 de enero de 2012. [Citado el: 27 de marzo de 2013.] http://jummp.wordpress.com/2012/01/10/desarrollo-de-software-tarjetas-crc/.