Universidad de las Ciencias Informáticas

Facultad 1

MÓDULO DE IMPLEMENTACIÓN DE SENTENCIAS DE MANIPULACIÓN DE BASES DE DATOS RELACIONALES PARA LA PLATAFORMA RDB-LEARNING

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: Ortelio Francisco Hernández Socarrás

Tutores:

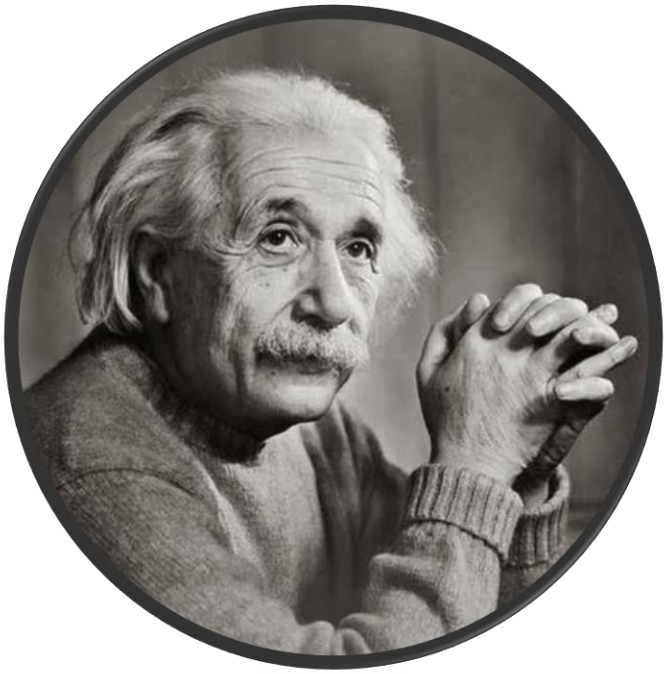
Ing. Yaniel Lázaro Aragón Barreda

MSc. Eylin Hernández Luque

Ing. Yarileidis Barcena Calzado

La Habana, junio 2017

"Año 59 de la Revolución"



Todos somos unos genios pero si juzgamos a un pez por su habilidad de escalar un árbol, vivirá la vida entera creyendo que es un estúpido

Albert Einstein

DECLARACIÓN JURADA DE AUTORÍA

Declaro por este medio que yo Ortelio Francisco Hernández Socarrás, con carné de identidad 92060440024 soy el autor principal del trabajo de diploma titulado *"Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma RDB-Learning*" y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste se firma la presente, a los ___ días del mes de _____ del año 2017.

Autor
Ortelio Francisco Hernández Socarrás

Tutor
Ing. Yaniel Lázaro Aragón Barreda

Tutor
MSc. Eylin Hernández Luque

Tutor
Ing. Yarileidis Barcena Calzado

DEDICATORIA

A mi familia, y principalmente a mis **hermanos** que han velado día a día mi crecimiento y formación como persona de bien.

A mi otra familia, la de mi novia **Claudia**, que durante casi 4 años me han acogido como un miembro más y me han apoyado en todo.

A mi segunda madre, la profe **María Elena Olivera** por sus consejos, educación y confianza.

Muy especialmente, a mi **MADRE** por ser la razón de mi existir, siempre querer lo mejor para mí y apoyarme en cada decisión a tomar para mi futuro como profesional.

En fin, a todos lo que han hecho posible que hoy me gradué como Ingeniero en Ciencias Informáticas.

A ustedes, con todo mi corazón.

Ortelio Francisco Hernández Socarrás

AGRADECIMIENTOS

Desde muy pequeño soñé con ser un profesional, hoy este sueño se hace realidad. Muchos años han pasado para poder construirlo y siempre han estado presente el esfuerzo, la dedicación y el sacrificio. Es por eso, que quiero agradecer en un principio a mi **país**, a la **Revolución** cubana y al eterno **Comandante en Jefe Fidel Castro Ruz**, creador de esta Universidad que me acogió y me dio la posibilidad de formarme como profesional.

Agradecer también a cada una de las personas que de una forma u otra han aportado su granito de arena en mi formación durante estos cinco años. A todos mis profesores y compañeros que me han ayudado a alcanzar mis metas y resultados docentes, pudiendo así hoy optar por un **Título de ORO**. Al tribunal que hoy es testigo del profesional que se forma, las profesoras **Niurvis** y **Cecilia,** así como los profesores **Noichel** e **Israel**, a ustedes gracias.

A la primera familia que encontré al llegar a la UCI, mis compañeros de aula. Cada uno con su peculiar manera de ser, de una forma u otra me han enseñado algo. A **Aimet**, la primera amiga que encontré en la universidad, gracias por soportarme durante todo este tiempo. A **Daynis** mi compañera de puesto gracias por tu apoyo. A **Arlene**, **María**, **Leydis**, **Mariam**, **Rosalba**, **Yesenia** a todas mil gracias por su cariño y este tiempo juntos. De los varones, primeramente, agradecer a mi cuasi-compañero de tesis **Carlos Yordan**, a mi compañero y amigo de andanzas **Osmel**, a mi compañero de cuarto **Roly**, a **Lázaro**, **Ignacio**, **Abel**, **Alejandro**, **Leonard** y a quién ya no nos acompaña hoy, **César**, por ser el alma de diversión del aula, siempre quedándose dormido y cuando no haciendo sus chistes pesa 'os, a ti mi hermano te deseo éxitos.

A mis tutores por todo su apoyo: **Yari** por ser mi guía, **Eylin** por aportarme su conocimiento científico y a **Yaniel** por la confianza depositada en mi para realizar junto a él, este proyecto. Agradezco de igual manera, aunque no es tutor oficial, siempre dedicó un pedacito de su tiempo para atenderme, al profe **Julio César Espronceda**, llegue a usted también mi agradecimiento.

De igual manera agradecer a la **FEU** organización de la cual fui dirigente. Esta me enseñó virtudes como la responsabilidad y el sacrificio. Me posibilitó conocer a muchos amigos, así como compañeros de trabajo. A **Joshy, Yosvany, Manuel, Yojahny, Sandra, Elianis, Lester, Niurka, Leyriel, Arlenis, Juan Gabriel**, en fin, a toda la familia FEU, gracias. A los sobrinos a la fuerza que me he buscado aquí en la UCI, **Luis** y **Beatriz** a ustedes igual les agradezco.

A mis amigos **Esteban, Dayan, Aneylis, Greter, Lidice, Ernesto** por compartir conmigo durante este tiempo y brindarme su amistad incondicional. A **Chabelly, Andy, Samanda** que a pesar del poco tiempo que nos conocemos me han bridado su confianza, apoyo y ánimo.

De igual manera agradezco a la familia que he formado en la UCI, a mi prima de apellido y de cariño **Tania**, a su novio y mi colega **Eliuvis**, a mi amigo de mucho tiempo **José Angel**. Agradezco también a una persona que ha podido sopórtame durante casi cuatro años y gracias a ella mi vida ha cambiado completamente. Gracias, **Claudia Rafaela**, por tu paciencia, dedicación, cariño, preocupación y no sé cuántas cosas más. Gracias por llegar a mi vida.

También agradezco a la familia de mi novia que me ha acogido como un miembro más, apoyándome con lo que fuese posible. A mis suegros, casi padres **Teresa** y **Eulicer**, a mi cuñado, casi hermano **Ulises.** Así como a los demás miembros de la familia. A todos ustedes gracias.

De manera especial quiero agradezco a mi familia. A mi **padre**, mis hermanos **Yoanis** y **Coqui** y a quien siempre será como otro padre para mi **Alberto Nápoles (Beto),** a todos por siempre velar por mi crecimiento y formación como profesional. El mejor de los agradecimientos va dirigido al ser que me dio la vida, por ser la razón de mi existir, mi **MADRE**, por siempre querer los mejor para mi y apoyarme en cada decisión. A ti mil **GRACIAS**.

A las amistades que por alguna razón he conocido, pero no dudan en saludar, preguntarme como me va y se encuentran aquí presentes. Todos y cada uno de ustedes reciban igual hoy mi agradecimiento.

Ortelio Francisco Hernández Socarrás

RESUMEN

La Universidad de las Ciencias Informáticas es un centro de estudios superiores caracterizado por el uso de las Tecnologías del Aprendizaje y el Conocimiento en el proceso docente-educativo. Como parte del plan de estudio de la carrera Ingeniería en Ciencias Informáticas, se imparte la asignatura Sistema de Bases de Datos I, la cual cuenta con dos habilidades genéricas necesarias para cumplir con los objetivos de la misma. Una de estas es implementar sentencias de manipulación de Bases de Datos Relacionales. Las herramientas empleadas en la asignatura y la plataforma en desarrollo *RDB-Learning,* actualmente, no permiten la correcta implementación de este tipo de sentencias. La presente investigación presenta un módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*, el cual permite solucionar ejercicios de implementación a partir de una interfaz gráfica para guiar la construcción de las sentencias. El funcionamiento de este módulo se basa en la tecnología *e-learning*, al permitir la retroalimentación de sus usuarios a través de la plataforma. Se presenta una estrategia de pruebas realizada al módulo, con el objetivo de entregar al cliente una solución confiable y libre, que pueda ser empleada como herramienta de apoyo a la asignatura.

Palabras claves: bases de datos relacionales, *e-learning*, implementación, sentencias.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) han tenido un auge en su desarrollo durante las últimas décadas. En la sociedad del conocimiento, estas desempeñan un papel importante en el desarrollo en todos los sectores, como es el caso de la educación. En este sentido, los centros educacionales deben adaptarse a los nuevos tiempos y a las posibilidades que las TIC les ofrecen para ampliar la oferta educativa: crear entornos flexibles para el aprendizaje, incrementar las modalidades de aprendizaje y potenciar el uso de los entornos interactivos (Cabero-Almenara, 2004).

La introducción de estas ventajas en el entorno educativo y la necesidad de vincular aún más las TIC a la Universidad del siglo XXI, han conllevado a la aparición del concepto de Tecnología del Aprendizaje y del Conocimiento (TAC), la cual es la fusión entre tecnología y metodología. En ese sentido, se define a las TAC como un mecanismo para tratar de orientar hacia un uso más formativo de la tecnología, tanto para el estudiante, como para el profesor, con el objetivo de lograr la calidad en el proceso de aprendizaje (Granados-Romero, 2015).

Según Lozano (2011), las TAC tratan de incidir en la metodología, el uso de la tecnología y no de asegurar el dominio de unas cuantas herramientas informáticas. Se trata, en definitiva, de conocer y explorar los posibles usos didácticos que tienen las TIC para el aprendizaje y la docencia. Es decir, las TAC van más allá de aprender a usar las TIC, y apuestan por explorar estas herramientas tecnológicas al servicio del aprendizaje y de la adquisición de conocimientos. Las mismas implican aprender a aprender por medio de recursos virtuales de aprendizaje.

Este proceso, también conocido como aprendizaje electrónico o *e-learning*, es una modalidad del proceso de enseñanza-aprendizaje que consiste en el diseño, puesta en práctica y evaluación de un curso o plan formativo desarrollado a través de la red de computadoras. Puede definirse como una educación o formación ofrecida a individuos que están geográficamente dispersos o separados o que interactúan en tiempos diferidos del docente, empleando los recursos informáticos y de telecomunicaciones. La principal característica del *e-learning* es que todo el proceso formativo tiene lugar, totalmente o en parte, por medio de una especie de aula o entorno virtual en el que interactúan estudiantes y profesores. El empleo de esta técnica ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional. Es por eso que actualmente ocupa un lugar destacado y reconocido dentro de las organizaciones empresariales y educativas (e-ABC, 2017).

Tal es el caso de la Educación Superior en Cuba, la cual juega un destacado papel en la formación de

profesionales, la superación permanente y el uso de las tecnologías *e-learning* en el proceso docente-educativo. La vinculación de estas tecnologías en dicho proceso es considerada un reto para los docentes, sin embargo, desde el punto de vista de los estudiantes resulta más atractiva. Esta permite innovar y mejorar los métodos tradicionales de la enseñanza presencial y brinda, además, diversas modalidades para realizar estudios superiores, ya sea completamente a distancia o mixto.

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios superiores que se caracteriza por el uso de las TAC en el proceso docente-educativo. Desde su creación, esta desarrolla un continuo perfeccionamiento en aras de cumplir con su misión estratégica: formar profesionales comprometidos con la Revolución y altamente calificados en la rama de la informática y en la producción de software y servicios.

Desde su Modelo de Formación del Profesional (Ministerio de Educación Superior [MES], 2014), se señala la necesidad de una docencia dirigida a la auto-preparación de los estudiantes haciendo uso de las tecnologías. De ahí que la institución potencie el uso de la Plataforma Educativa ZERA, el Repositorio de Objetos de Aprendizaje RHODA y la producción de medios tecnológicos de apoyo al proceso de enseñanza-aprendizaje.

En función de cumplir la misión encomendada, la UCI cuenta con un Plan de Estudio para la carrera Ingeniería en Ciencias Informáticas (ICI), dentro del cual incluye la asignatura Sistemas de Bases de Datos I (SBD I), correspondiente a la disciplina Ingeniería y Gestión de Software. En la misma se trabajan dos habilidades genéricas necesarias para vencer sus objetivos. Una de estas es implementar sentencias de manipulación de Bases de Datos Relacionales (BDR), habilidad que se refiere a garantizar el tratamiento de la información contenida en un Sistema de Bases de Datos Relacional (MES, 2014).

Como soporte para la impartición de esta asignatura se emplean algunas herramientas, siendo la plataforma ZERA una de ellas, en la misma están a disposición de los estudiantes, diversos recursos que pueden ser consultados para el estudio, entre los que se pueden destacar: materiales bibliográficos, cuestionarios y ejercicios para la implementación de sentencias de manipulación de BDR.

En esta plataforma no se brinda soporte a la resolución de los ejercicios antes mencionados, por lo cual es necesario emplear una herramienta externa. Esta herramienta ofrece la posibilidad de implementar sentencias manualmente y haciendo uso de una interfaz gráfica. Sin embargo, a través de dicha interfaz, solo es posible implementar las sentencias de selección (*SELECT*) de forma parcial, pues no tiene en cuenta todas las cláusulas de esta. De igual manera, no se permite la creación de todas las sentencias pertenecientes al lenguaje de manipulación de datos, como son: la inserción (*INSERT*), modificación

(*UPDATE*) y eliminación (*DELETE*). Esto trae consigo que se dificulte la resolución de algunos de los ejercicios propuestos en la asignatura. Por otra parte, esta herramienta, según su entorno de ejecución, se considera una aplicación de escritorio que, además, no está basada en tecnología *e-learning*; lo cual impide el desarrollo de una auto-preparación de los estudiantes que sea guiada, controlada y evaluada por el profesor desde espacios de lugar y tiempo diferentes, así como la retroalimentación entre profesores y estudiantes.

Basado en las TAC y con el objetivo de servir de soporte tecnológico al proceso de enseñanza-aprendizaje de la asignatura SBD I, en la Facultad 1 de la UCI, se encuentra en desarrollo la Plataforma educativa *RDB-Learning*. Actualmente, la misma no permite la implementación de sentencias de manipulación de BDR para la resolución de ejercicios propuestos en la asignatura, deficiencia por la cual aún no puede ser empleada como herramienta de apoyo a la misma.

Teniendo en cuenta la problemática anteriormente descrita, se identificó como **problema de investigación:** ¿Cómo contribuir, mediante el uso de las tecnologías del aprendizaje y el conocimiento, a la implementación de sentencias de manipulación bases de datos relacionales en la asignatura Sistemas de Bases de Datos I en la Facultad 1 de la Universidad de las Ciencias Informáticas?

El **objeto de estudio** de la presente investigación va enfocado a las herramientas informáticas para manipulación de bases de datos relacionales.

Para dar solución al problema planteado se define como **objetivo general:** desarrollar un módulo como tecnología del aprendizaje y el conocimiento para la Plataforma *RDB-Learning* que permita la implementación de sentencias de manipulación de bases de datos relacionales en la asignatura Sistemas de Bases de Datos I en la Facultad 1 de la Universidad de las Ciencias Informáticas.

Enmarcado en el **campo de acción:** las tecnologías del aprendizaje y el conocimiento para la implementación de sentencias de manipulación de Bases de Datos Relacionales en la asignatura Sistemas de Bases de Datos I en la Facultad 1 de la Universidad de las Ciencias Informáticas.

En aras de cumplir el objetivo general definido se plantean las siguientes **preguntas científicas:**

- ¿Cuáles son los referentes teóricos que sustentan el desarrollo de la investigación, relacionados con el empleo de las herramientas informáticas en función de la implementación de sentencias de manipulación de bases de datos relacionales?
- ¿Qué características tienen las herramientas informáticas existentes para la implementación de sentencias de manipulación de bases de datos relacionales?

- ¿Qué elementos deben tenerse en cuenta para realizar el análisis y diseño del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*?
- ¿Cómo llevar a cabo el proceso de desarrollo del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning,* a partir de los diseños especificados?
- ¿Qué resultados se obtendrán al validar, a través de una estrategia de pruebas de software, el módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*?

Acorde a las preguntas científicas expuestas, se proponen las siguientes **tareas de investigación**:

- Sistematización de los referentes teóricos que sustentan el desarrollo de la investigación relacionados con el empleo de las herramientas informáticas en función de la implementación de sentencias de manipulación de bases de datos relacionales.
- Análisis de las características de las herramientas existentes para la implementación de sentencias de manipulación de bases de datos relacionales.
- Análisis y diseño de las funcionalidades del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*.
- Implementación de las funcionalidades del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*.
- Validación, mediante una estrategia de pruebas de software, del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*.

Para el desarrollo de las tareas fueron empleados los siguientes **métodos de investigación**:

Métodos teóricos:

Histórico-Lógico: se emplea para identificar posibles funcionalidades que pueda tener el módulo a partir del análisis de la evolución de las herramientas informáticas para la implementación de sentencias de manipulación de BDR.

Analítico-Sintético: empleado para el análisis, evaluación y selección de las técnicas a emplear en el desarrollo del módulo. Así como para sintetizar la información que se obtuvo mediante la entrevista con el cliente de manera que pudiera ser usada en el desarrollo del mismo, además, en la identificación de

los elementos del marco teórico de la investigación.

Modelación: para realizar una representación del proceso estudiado que sirva de guía en el desarrollo del módulo, y mediante este, identificar las características y relaciones fundamentales.

Métodos empíricos:

Entrevista: empleado en los encuentros con el cliente para obtener la información necesaria que permita determinar las características, cualidades y requisitos con los que debe contar la propuesta de solución.

Observación: empleado para obtener el conocimiento necesario del funcionamiento de las soluciones existentes para la manipulación de BDR.

Estructura de la investigación

El presente trabajo de diploma está estructurado de la siguiente manera: introducción, tres capítulos, conclusiones generales, recomendaciones y referencias bibliográficas empleadas durante el desarrollo de la investigación, y por último para complementar la investigación se presentan una serie de anexos.

En el **capítulo 1** *"Fundamentos teóricos de la investigación"* se abordan aspectos teóricos que apoyan la investigación y se definen los principales conceptos. Se analiza la sintaxis de las sentencias de manipulación de BDR. Además, se analizan algunas herramientas para la implementación de sentencias de manipulación de BDR. Se define el entorno de desarrollo de la propuesta de solución.

El **capítulo 2** *"Características y diseño del Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma RDB-Learning"* se analiza cada uno de los artefactos y elementos para el diseño de la solución, que permiten una mejor comprensión de las funcionalidades a desarrollar. Además, se expone las principales características de las herramientas, su diseño, estilo arquitectónico, y requisitos a tener en cuenta en la implementación.

En el **capítulo 3** *"Implementación y pruebas del Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma RDB-Learning"* se define el estándar de codificación que sirve de guía para la implementación de la solución propuesta, así como la estrategia de pruebas a aplicar.

CAPÍTULO 1 “Fundamentos teóricos de la investigación”

1.1. Introducción

Durante una investigación es necesario obtener el conocimiento teórico que facilite la realización de las actividades y así poder llevar a la práctica lo aprendido. Para ello se hace necesario realizar una profunda búsqueda bibliográfica con el fin de lograr una mayor comprensión del alcance de la investigación. En este capítulo se abordan los elementos teóricos que fundamentan la presente investigación. Se realiza un análisis de las herramientas existentes que permiten la manipulación de BDR. Se incluye además un análisis de la metodología, herramientas y tecnologías a emplear en el desarrollo de la propuesta de solución.

1.2. Conceptos asociados a la investigación

Para ayudar a entender el desarrollo de la investigación, se relacionarán a continuación los conceptos principales que sirven de soporte.

Una **aplicación web**, se define como un sistema de hipermedia[1] donde los recursos se encuentran vinculados unos a otros, por lo que debe verse como un sistema de nodos interconectados a través de vínculos. Estos vínculos proporcionan la forma para navegar entre los recursos de la aplicación (Martínez, Guzmán, Alarcón, & Gómez, 2013).

Según Ávila Barrientos (2014), una aplicación web es un programa o conjunto de programas informáticos que ayudan a los usuarios de una computadora a procesar una tarea específica. Una aplicación web está compuesta por una interfaz de usuario y es accesible a través de un navegador web.

En la fase de diseño del ciclo de vida de una aplicación web, dado un problema a resolver, en primer lugar, hay que estudiar la posibilidad de dividirlo en otros más pequeños, llamados sub-problemas. Del mismo modo, también puede ser conveniente fragmentar estos sub-problemas obtenidos, hasta llegar a otros realmente sencillos. A cada sub-problema se le considera parte o módulo del problema global (Rodríguez, 2012).

Según Gallardo-Ruiz y García-López (2017), en programación, un **módulo** es un fragmento de un programa que se desarrolla de forma independiente del resto de la solución. Esta independencia hace posible un mecanismo de compilación por separado que limita la complejidad del programa que se está

[1] Se incluye no sólo texto, sino también otros medios: imágenes, audio, vídeo.

desarrollando. Al compilarse un módulo por separado, el desarrollador solo debe preocuparse de él, prescindiendo en parte de cómo se utiliza este dentro del programa en general. Quien escriba el resto del programa no debe preocuparse de los detalles del módulo sino sólo de cómo utilizarlo.

En el contexto de la investigación se hará referencia a las aplicaciones web con fines educativos o de enseñanza en línea (*e-learning*).

El término ***e-learning*** consiste en la educación y capacitación a través de Internet. Este tipo de enseñanza permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas (e-ABC, 2017).

Este nuevo concepto educativo, posibilitado por Internet, se posiciona como la forma de capacitación predominante. Esta tecnología ha transformado la educación, abriendo puertas al aprendizaje individual y organizacional. Es por ello que ha ocupado un lugar destacado dentro de las organizaciones empresariales y educativas (e-ABC, 2017).

El término "*e-learning*" es la simplificación de *Electronic Learning*. El mismo reúne a las diferentes tecnologías, y a los aspectos pedagógicos de la enseñanza y el aprendizaje. Entre los beneficios del *e-learning* se encuentran (e-ABC, 2017):

- **Reducción de costos**: permite reducir y hasta eliminar gastos de traslado, alojamiento, material didáctico
- **Rapidez y agilidad**: Las comunicaciones a través de sistemas en la red confiere rapidez y agilidad a las comunicaciones
- **Acceso *just-in-time***: los usuarios pueden acceder al contenido desde cualquier conexión a Internet, cuando les surge la necesidad
- **Flexibilidad de la agenda**: no se requiere que un grupo de personas coincidan en tiempo y espacio

Una **plataforma *e-learning***, es un espacio virtual de aprendizaje orientado a facilitar la experiencia de capacitación a distancia, tanto para empresas como para instituciones educativas. Este sistema permite la creación de "aulas virtuales"; en ellas se produce la interacción entre tutores y alumnos, y entre los mismos alumnos; como también la realización de evaluaciones, el intercambio de archivos, la participación en foros, chats, y una amplia gama de herramientas adicionales. Entre los beneficios de una plataforma de *e-learning* se encuentran (e-ABC, 2017):

- Brinda capacitación flexible y económica

- Combina el poder de Internet con el de las herramientas tecnológicas
- Anula las distancias geográficas y temporales
- Permite utilizar la plataforma con mínimos conocimientos
- Posibilita un aprendizaje constante y nutrido a través de la interacción entre tutores y alumnos
- Ofrece libertad en cuanto al tiempo y ritmo de aprendizaje

En otro sentido y relacionado de igual manera con la presente investigación, se encuentra el concepto de **base de datos** (BD), el cual se refiere a un conjunto de datos relacionados entre sí con un objetivo común. De acuerdo con C. J. Date, en su libro Introducción a las bases de datos (como se citó en Zaragosa, 2013):

> Es una colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de dato, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.

De igual manera Rouse (2014), define a una base de datos como una colección de información que está organizada para que pueda ser de fácil acceso, administrada y actualizada. Los datos se organizan en filas, columnas y tablas, y se indexan para facilitar la búsqueda de información relevante. De igual manera se actualizan, expanden y eliminan a medida que se agrega nueva información. Las bases de datos procesan cargas de trabajo para crear y actualizarse, consultar los datos que contienen y ejecutar aplicaciones sobre ella.

Una de las formas de manipulación de las BD es a través de **sentencias**, estas se definen como: las unidades ejecutables más pequeñas de un programa, las cuales especifican y controlan el flujo y orden de ejecución del mismo (Delgado, 2011). Existen tres categorías que subdividen a las sentencias *SQL*[2]: Lenguaje de Control de Datos, Lenguaje de Definición de Datos y Lenguaje de Manipulación de Datos

[2] Corresponde a la expresión inglesa **Structured Query Language** (entendida en español como **Lenguaje de Consulta Estructurado**) se identifica a un tipo de lenguaje vinculado con la gestión de bases de datos relacionales, que permite la especificación de distintas clases de operaciones entre éstas (Pérez Porto & Gardey, 2012).

(*DML*).

La presente investigación contempla el diseño de las sentencias *DML*: *SELECT, UPDATE, INSERT* y *DELETE*, por lo que se hace necesario establecer las sintaxis correspondientes a cada una.

1.3. Sintaxis para el empleo de sentencias de lenguaje de manipulación de datos

Lenguaje de Manipulación de Datos (*Data Manipulation Language, DML*), es un lenguaje proporcionado por los sistemas gestores de bases de datos, que permite a los usuarios de estos llevar a cabo las tareas de consulta o modificación de los datos contenidos en las bases de datos (Ortíz, 2015). En este sentido se incluyen sentencias para la recuperación, inserción, actualización y eliminación de datos.

Recuperación de datos (*SELECT*)

La recuperación de los datos en el lenguaje *SQL* se realiza mediante la sentencia *SELECT*. Esta sentencia permite indicar al SGBD, la información que se quiere recuperar. Esta sentencia *SQL* consta de:

- La cláusula ***SELECT,*** seguida de la descripción de lo que se desea ver, los nombres de las columnas a seleccionar (obligatorio)
- La cláusula ***FROM,*** seguida de la especificación de las tablas que contienen los datos (obligatorio)
- La cláusula ***WHERE,*** seguida por criterios de selección o condiciones (opcional)
- La cláusula ***GROUP BY,*** seguida de las columnas por las que se quiere agrupar los datos (opcional)
- La cláusula ***HAVING*****,** seguida de condiciones para los datos agrupados (opcional)
- La cláusula ***ORDER BY,*** seguida por el criterio de ordenación (opcional)
- La cláusula ***LIMIT*****,** seguida del número de registros a limitar (opcional)

De manera general su sintaxis es como la que se muestra a continuación.

```
SELECT [ALL|DISTINCT] <nombre_campo> [{, nombre_campo }]
[{,<función_agregacion>}]
FROM <nombre_tabla>|<nombre_vista>[{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condición> [{AND|OR <condición>}]]
[GROUP BY <nombe_campo> [{, <nombre_campo>}]]
[HAVING <condición> [{AND|OR <condición>}]]
[ORDER BY <nombre_campo>|<campo_indice> [ASC|DESC]
[{,<nombre_campo>|<campo_indice> [ASC|DESC]}]]
[LIMIT <cantidad_registros>]
```

Figura 1: Sintaxis de la sentencia SELECT

Inserción de datos (*INSERT*)

La inserción de datos en el lenguaje *SQL* se realiza mediante la sentencia *INSERT.* Esta sentencia permite indicarle al SGBD que se desea insertar nuevos registros en una tabla de la base de datos. De igual manera esta sentencia puede ejecutarse sobre vistas, siempre que cumplan ciertas condiciones (se hayan definido disparadores que añadan esos valores en las tablas sobre las que están definida la vista).

La sintaxis de esta sentencia es como se muestra a continuación:

```
INSERT INTO <nombre_tabla>|<nombre_vista> [({<nombre_campo>,})]
VALUES ({<valor_campo>,})
```

Figura 2: Sintaxis de la sentencia INSERT

Para insertar datos en una relación, se específica la tupla que se desea insertar o se formula una consulta cuyo resultado sea el conjunto de tuplas que se desea insertar. Los valores de los atributos de las tuplas que se inserten deben pertenecer al dominio de los atributos de la tabla.

Modificación de datos (*UPDATE*)

La sentencia *UPDATE* es usada para modificar los valores en columnas existentes de una tabla de la base de datos. Esta sentencia se aplica sobre una única tabla o vista (siempre que sea actualizable).

La sintaxis de esta sentencia es la siguiente:

```
UPDATE <nombre_tabla>|<nombre_vista>
SET {<nombre_campo> = <expresión>,}
[WHERE <condición> [{AND|OR <condición>}]]
```

Figura 3: Sintaxis de la sentencia UPDATE

Se especificará en la cláusula *SET* las columnas con los respectivos valores con que se actualizarán (si se desea actualizar a nulos, se asignará el valor *NULL*). La cláusula *WHERE*, indica las filas con las que se va a trabajar, si esta cláusula no se define, la actualización afectará a todas las filas de la tabla.

Eliminación de datos (*DELETE*)

Para la eliminación de los datos, en *SQL* se define la sentencia *DELETE*. Sentencia que permite eliminar una o varias filas de una única tabla.

La sintaxis de esta sentencia es la que sigue:

```
DELETE FROM <nombre_tabla>|<nombre_vista>
[WHERE <condición> [{AND|OR <condición>}]]
```

Figura 4: Sintaxis de la sentencia DELETE

Cuando se trabaja con esa sentencia se debe tener en cuenta las siguientes consideraciones:

- Solo se puede borrar datos de una única tabla
- Cuando se borra datos de una vista, se borran también de la tabla (las vistas son solo una forma de ver los datos, no una copia)
- Si se intenta borrar un registro de una tabla referenciada por una llave foránea (*FOREING KEY*) como tabla maestra, si la tabla dependiente tiene registros relacionados, la sentencia *DELETE* fallará a no ser que la tabla hija contenga la sentencia *ON DELETE CASCADE*.

1.4. Análisis de las soluciones existentes

Actualmente existen diversas soluciones para el manejo de BDR. Pero en el ámbito de la presente investigación, se tomaron en cuenta solamente aquellas que permitieran implementar sentencias a partir de una interfaz gráfica. A continuación, se relacionan sus características y principales funcionalidades.

CAPÍTULO 1

Administración de PostgreSQL *pgAdmin III*

PgAdmin III es la herramienta *Open Source*[3] de administración por excelencia para las bases de datos *PostgreSQL*. Algunas de sus características son: el soporte completo para *UNICODE*[4], edición rápida de consultas, datos multihilo y soporte para todos los tipos de objetos de *PostgreSQL*. Es el centro de administración para las bases de datos *PostgreSQL*. Incluye una interfaz gráfica de administración, una herramienta para el trabajo con SQL, un editor de código de procedimientos y funciones y un constructor gráfico de consultas (ver Figura 5). Está disponible en más de 30 lenguajes y para varios sistemas operativos (pgAdmin, 2016).

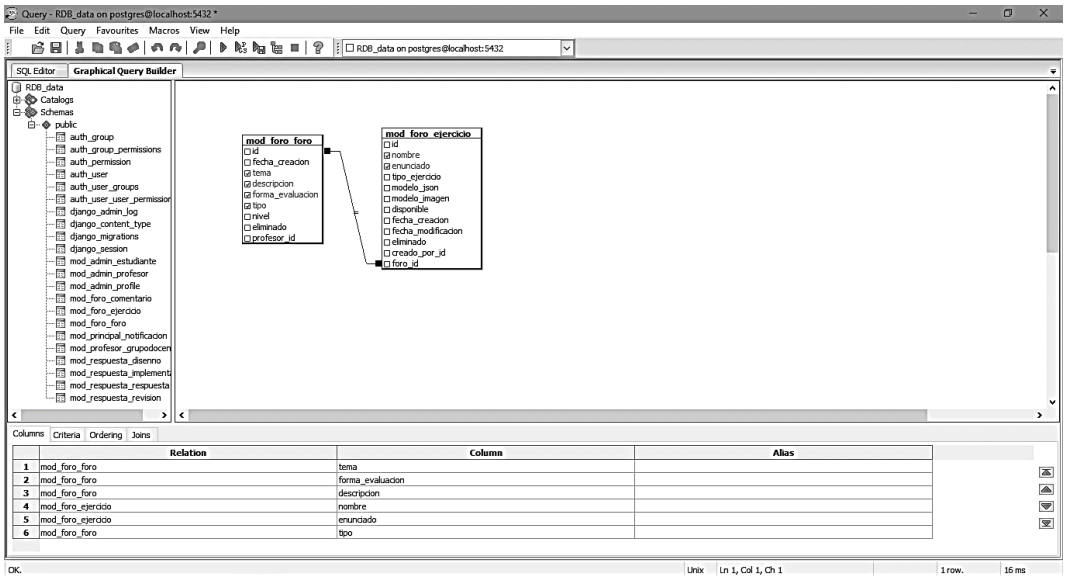


Figura 5: Interfaz del Grafical Query Builder de pgAdmin III

Navicat *Premium*

Navicat es un administrador gráfico de base de datos y un software de desarrollo producido por *PremiumSoft™ CyberTech Ltd*. para *MySQL, MariaDB, Oracle, SQLite, PostgreSQL* y *Microsoft SQL Server*. Está basado en el tipo de licencia Propietario/*Shareware*. Además, de que cuenta con un explorador como interfaz gráfica de usuario, en el que soporta múltiples conexiones para bases de datos locales y remotas. Su diseño está pensado para satisfacer las diferentes necesidades de sus usuarios (PremiumSoft™ CyberTech Ltd, 2017).

[3] Es el término con el que se conoce al software distribuido y desarrollado libremente.
[4] Es un estándar en el que se definen todos los caracteres necesarios para la escritura de la mayoría de los idiomas hablados en la actualidad que se usan en la computación.

Las características de *Navicat* incluyen (PremiumSoft™ CyberTech Ltd, 2017).:

- Visualizador del generador de consultas (ver Figura 6)
- Túnel *SSH* y *HTTP*
- Migración y sincronización de datos y de estructuras
- Importación, exportación y copia de seguridad de datos
- Generador de Informes
- Programador de tareas y herramientas para asistentes

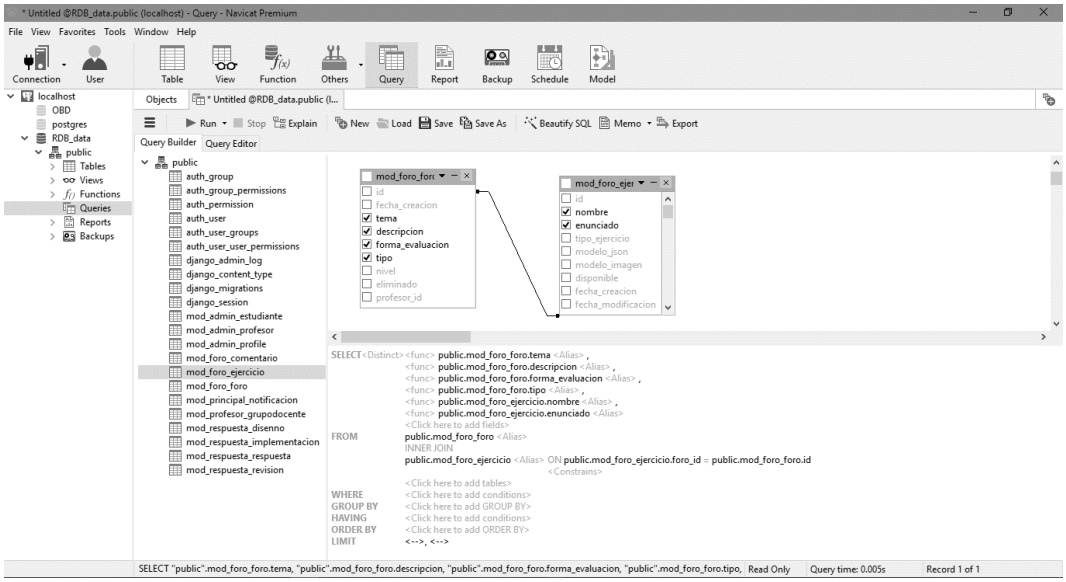


Figura 6: Interfaz del Query Builder de Navicat Premium

Visual SQL Builder

Visual SQL Builder es una herramienta escrita en *Java*. La cual permite construir de forma visual sentencias SQL para la base de datos *PostgreSQL* sin la necesidad de escribir ninguna cláusula de la misma (ver Figura 7). Esta aplicación puede ayudar a personas con conocimientos básicos de informática a construir sentencias *SQL* con poco esfuerzo y entrenamiento, reduciendo los errores sintácticos y lógicos, y el tiempo de depuración de los mismos (Ochoa P, 2008).

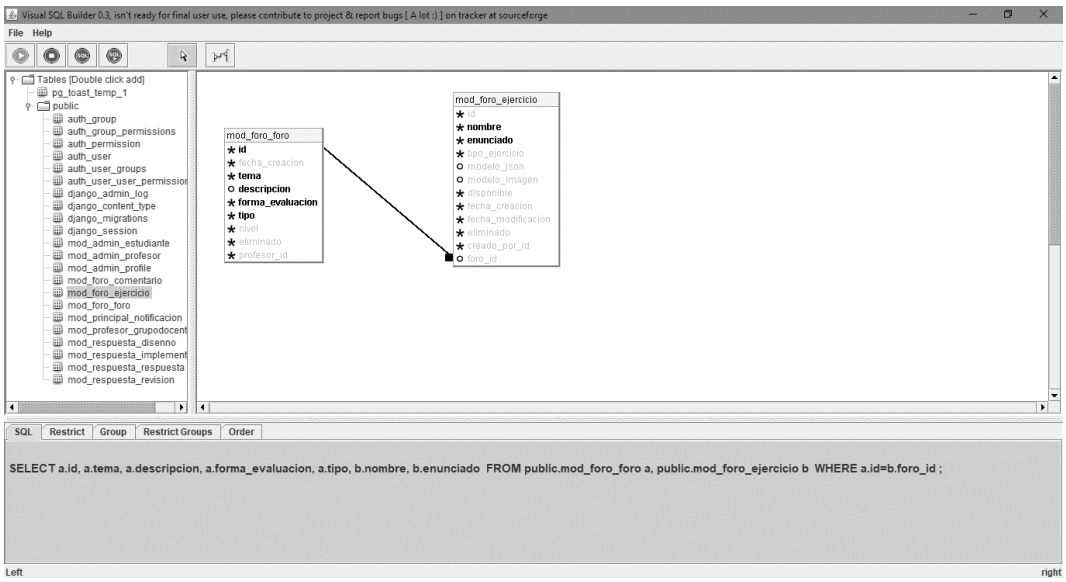


Figura 7: Interfaz del Visual SQL Builder (Fernandez, 2013)

Administrador de *MySQL phpMyAdmin*

PhpMyAdmin es una herramienta de software libre escrita en PHP, destinada a manejar la administración de *MySQL* a través de la Web. *PhpMyAdmin* soporta una amplia gama de operaciones en *MySQL* y *MariaDB*. Así como soporte para la mayoría de las características de *MySQL*:

- Examinar y eliminar bases de datos, tablas, vistas, campos e índices
- Crear, copiar, eliminar, cambiar el nombre y alterar bases de datos, tablas, campos e índices
- Servidor de mantenimiento, bases de datos y tablas, con propuestas sobre la configuración del servidor
- Ejecutar, editar y marcar cualquier sentencia *SQL* (ver Figura 8), incluso *batch-queries*
- Administrar las cuentas de usuario y privilegios de *MySQL*
- Gestionar procedimientos almacenados y disparadores

Las operaciones de uso frecuente se pueden realizar a través de la interfaz de usuario, mientras que todavía tiene la capacidad de ejecutar directamente cualquier sentencia *SQL*. Se encuentra bajo la licencia *GPL* Versión 2 y disponible en 72 idiomas.

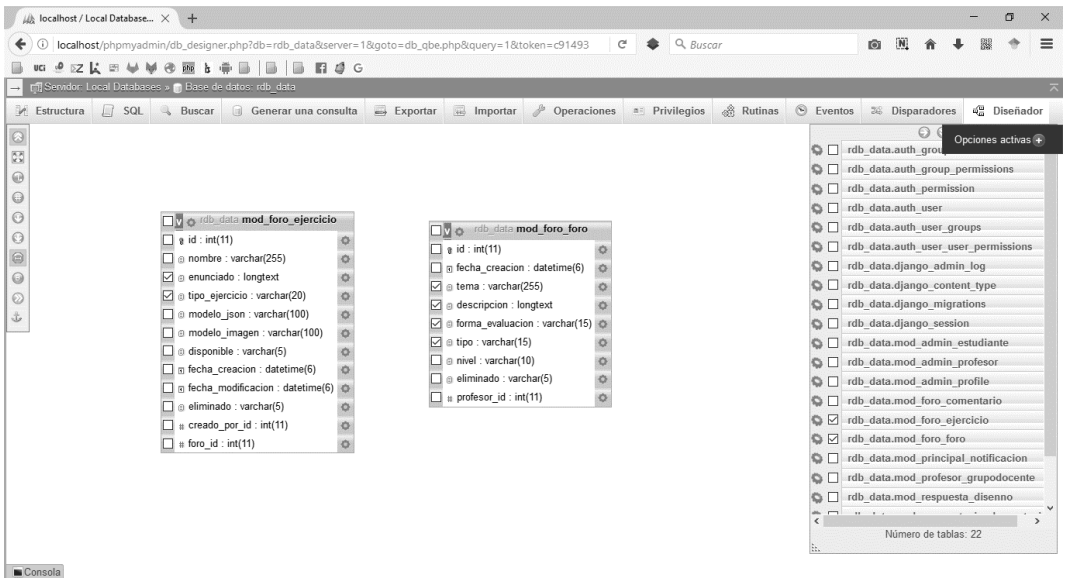


Figura 8: Interfaz Constructor Visual phpMyAdmin

EMS *SQL Query*

Es una herramienta utilitaria creada por la compañía de software EMS y permite crear consultas *SQL* hacia diversas bases de datos mediante el uso de una interfaz gráfica (ver Figura 9), que le permite conectarse a la base de datos, seleccionar tablas y campos para una consulta, fijar un criterio de selección de filas, entre otras opciones. Le permite al usuario trabajar con varias consultas de forma simultánea y ver los resultados de ejecución de las mismas en diferentes tipos de salida y dentro de estas, llevar a cabo la manipulación de los datos mostrados. Su tipo de licenciamiento es privativo y no posee código fuente disponible (EMS Database Management Solutions, Inc, 2017).

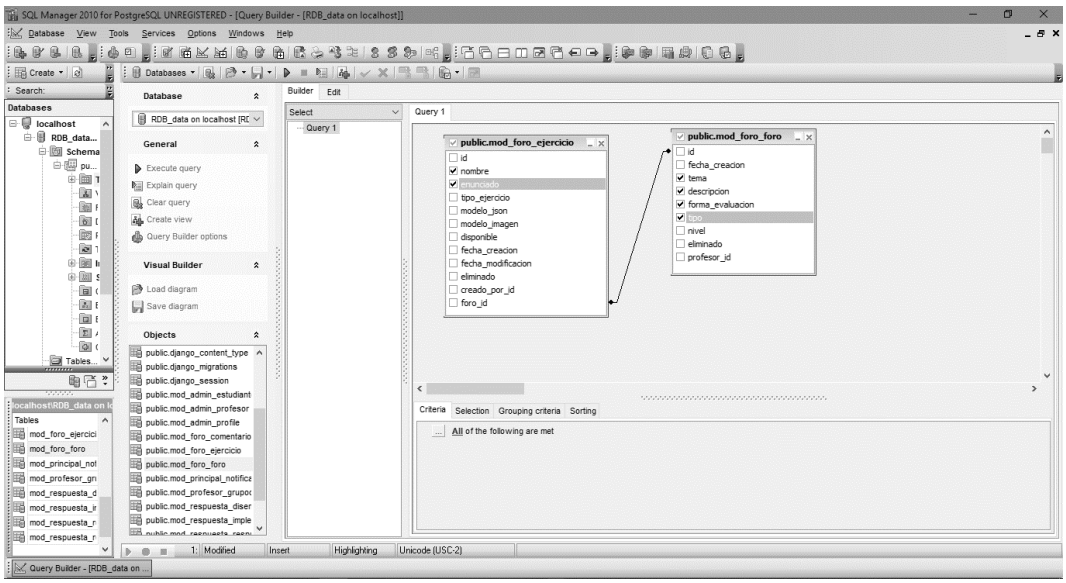


Figura 9: Interfaz del Query Builder de EMS SQL Query

Conclusiones del análisis de las soluciones existentes

Luego de realizado el análisis de las herramientas anteriores se concluye que, aun cuando cuentan con interfaz visual para la implementación de sentencias, en la mayoría de los casos esta no es muy intuitiva, ni agradable para el usuario. Por otra parte, el inconveniente de emplear estas soluciones en el proceso de enseñanza-aprendizaje de la asignatura específicamente para la implementación de sentencias de manipulación de BDR; radica que en la mayoría son aplicaciones de escritorio, por lo que ninguna hace uso de las potencialidades que brinda la tecnología *e-learning* con este objetivo. Sin embargo, estudiarlas posibilitó al autor la identificación de funcionalidades y tecnologías que pueden contribuir al desarrollo de la propuesta de solución de la investigación.

Tabla 1: Resumen de soluciones existentes analizadas

Herramienta	Licencia	Tipo Aplicación	E-learning
pgAdmin III	*PostgreSQL License*	Escritorio	NO
Navicat *Premium*	Privativo/*Shareware*	Escritorio	NO
Visual SQL Builder	*Dual* GPL / LGPL v2.	Escritorio	NO
phpMyAdmin	GPL v2.0	Web	NO
EMS *SQL Query*	Privativo	Escritorio	NO

Licencia: hace referencia al tipo de licencia de la aplicación.

Tipo de aplicación: se refiere a si aplicación de web o de escritorio.

E-learning: especifica si la aplicación es o forma parte de alguna plataforma educativa.

1.5. Entorno de desarrollo de la propuesta de solución

Teniendo en cuenta que el sistema a desarrollar es un módulo, este debe ser implementado en el mismo entorno de desarrollo del sistema al cual se debe integrar. Luego de analizar la documentación relacionada a la plataforma *RDB-Learning*, la cual documenta las herramientas, así como la metodología de desarrollo empleada, se especifica el uso del siguiente entorno de desarrollo.

1.5.1. Modelado de software

El modelado de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a "visualizar" el sistema a construir. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Por último, las herramientas de modelado y las de Ingeniería de Software Automatizada pueden ayudar a verificar la corrección del modelo (Academic, 2012).

Lenguaje Unificado de Modelación (*UML*)

Como lenguaje de modelado se utiliza el Lenguaje Unificado de Modelado (*UML* por sus siglas en inglés) en su versión 2.1. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Incluye aspectos conceptuales tales como procesos de negocio y funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (García, 2005).

Herramienta para el modelado

Para la presente investigación se utiliza la herramienta de modelado *Visual Paradigm* 8.0, es una herramienta de Ingeniería Asistida por Computadora (*CASE*[5] por sus siglas en inglés), que está diseñada para modelar un sistema de información empresarial y gestionar procesos de desarrollo. Esta herramienta soporta los principales lenguajes y estándares de modelado de la industria, en los que incluye el Lenguaje de Modelado Unificado (*UML*). Ofrece herramientas completas para la captura de requisitos de *software*, análisis de procesos, modelado de *software*, entre otras (Visual Paradigm, 2016).

1.5.2. Herramienta para control de versiones

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que se pueda recuperar versiones específicas más adelante (Saldaña, 2017).

Para garantizar el control de versiones en el sistema, y a su vez, la continua integración del módulo a la plataforma, se propone el uso de la versión 2.8 del cliente *Git* para la plataforma *Gitlab* disponible en la UCI en la dirección http://codecomunidades.prod.uci.cu, el cual, según Saldaña (2017):

- Es un sistema de control de versiones distribuido
- No depende de acceso a la red o un repositorio central
- Está enfocado a la velocidad, uso práctico y manejo de proyectos grandes

1.5.3. Lenguaje de Programación (lado del servidor)

Según (Hernandez Castillo, 2008), un lenguaje de programación consiste en un conjunto de órdenes o comandos que describen un proceso determinado. Cada lenguaje tiene sus instrucciones y enunciados verbales propios, que se combinan para formar los programas de cómputo. Los lenguajes de programación no son aplicaciones, sino herramientas que se pueden usar para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

En el desarrollo del módulo se emplea *Python* en su versión 2.7.11, el cual es un lenguaje de

[5] ***CASE*** (*Computer Aided Software Engineering*) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software (Marqués-Andrés, 2001).

programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, debido a que soporta orientación a objetos, programación imperativa y en menor medida, programación funcional. Es uno de los lenguajes de programación más populares en los últimos años debido la curva de aprendizaje (Rossum, 2017).

1.5.4. Marco de Trabajo

Un marco de trabajo o *framework* es una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Proporcionan un conjunto de librerías con funcionalidades que aumentan la facilidad del trabajo y disminuyen su complejidad. Además, permiten reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones (Euphoria IT, 2008). En la actualidad son muy usados para simplificar y agilizar el proceso de desarrollo de aplicaciones.

Para el desarrollo del módulo se utiliza el *framework* Django en su versión 1.9.5, que es un marco de trabajo de desarrollo web basado en el estilo arquitectónico *Model-View-Template* que solo cambia los nombres del estilo clásico Modelo-Vista-Controlador (MVC) y le otorga a las vistas toda la responsabilidad del negocio. El mismo está totalmente implementado sobre Python. Con él que se pueden crear y mantener aplicaciones de alta calidad e incluye un servidor web ligero que se puede usar mientras se desarrolla (Django, 2016).

1.5.5. Lenguaje de programación (lado del cliente)

En el desarrollo del módulo se utiliza el lenguaje de programación para el lado del cliente *JavaScript*. Este, es un lenguaje interpretado que permite incluir macros en páginas web. Estas macros se ejecutan en el ordenador del visitante, y no en el servidor, pues los servidores web suelen estar sobrecargados, mientras que las computadoras de los usuarios no suelen estarlo (Gutiérrez, 2009).

Librerías

Una librería es un conjunto de recursos (algoritmos) prefabricados, que pueden ser utilizados por el programador para realizar determinadas operaciones. Las declaraciones de las funciones utilizadas en estas librerías, junto con algunas macros y constantes predefinidas que facilitan su utilización, se agrupan en ficheros de nombres conocidos que suelen tener las extensiones *lib*, *bpl*, a, *dll*, *js*, entre otras (Torrealba, 2016). Las librerías utilizadas en el desarrollo del módulo son:

JQuery, es una librería multiplataforma de *JavaScript*, que permite simplificar la manera de interactuar

con los documentos *HTML*[6], manipular el árbol *DOM*[7], manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX*[8] a páginas web. Es además un software libre y de código abierto, permitiendo su uso en proyectos tanto libres como privados. *JQuery*, al igual que otras librerías, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta, se logran grandes resultados en menos tiempo y espacio (Gutiérrez, 2009).

GoJS, es una librería de *JavaScript* con múltiples funciones para implementar diagramas interactivos personalizados y visualizaciones complejas en navegadores y plataformas web modernos. Facilita la construcción de diagramas con nodos, enlaces y grupos complejos, con plantillas y diseños personalizables. Ofrece, además, muchas características avanzadas para la interactividad del usuario, tales como arrastrar y soltar, copiar y pegar, edición de texto en el lugar, información sobre herramientas, menús contextuales, diseños automáticos, plantillas, vinculación y modelos de datos, administración de estados transaccionales y deshacer, paletas, vistas generales, controladores de eventos, comandos y un sistema de herramientas extensible para operaciones personalizadas (GoJS, 2017).

GoJS es puro *JavaScript*, por lo que los usuarios obtienen interactividad sin necesidad de recorridos de ida y vuelta a servidores y sin complementos. Normalmente se ejecuta completamente en el navegador, renderizado a un elemento de lienzo *HTML5* sin requisitos de servidor. No depende de librerías o *frameworks JavaScript*, por lo que debería funcionar con cualquier *framework HTML* o *JavaScript* o sin estos (GoJS, 2017).

Formato de intercambio de datos

JSON, acrónimo de *JavaScript Object Notation*, es un formato de texto ligero para el intercambio de datos. Es un subconjunto de la notación literal de objetos de *JavaScript*, aunque hoy, debido a su amplia adopción como alternativa a *XML*, se considera un formato de lenguaje independiente. Una de las ventajas de *JSON* como formato de intercambio de datos es la sencillez para escribir un analizador sintáctico (*parser*) de *JSON*. En JavaScript, un texto *JSON* se puede analizar fácilmente usando la función *eval*(), lo cual ha sido fundamental para que *JSON* haya sido aceptado por parte de la comunidad

[6] Del inglés *HyperText Markup Language* (Lenguaje de marcas de hipertexto).
[7] Del inglés *Document Object Model* (Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos).
[8] Del inglés, acrónimo de *JavaScript* asíncrono y *XML*.

de desarrolladores *AJAX*, debido a la ubicuidad de *JavaScript* en casi cualquier navegador web (Gutiérrez, 2009).

1.5.6. Entorno Integrado de Desarrollo

Para agilizar la construcción de aplicaciones, los desarrolladores se apoyan de un Entorno de Desarrollo Integrado (*IDE* por sus siglas en inglés). Estos programas contienen un conjunto de herramientas de programación que permiten el desarrollo de otras aplicaciones en determinado lenguaje. Estos pueden realizar las funciones de un editor de código, un compilador, depurador y hasta un constructor de interfaz gráfica (Vigo, 2014)

En el desarrollo del módulo se emplea el *IDE Jetbrains PyCharm* en su versión 2016.3.2, el cual es un entorno de desarrollo integrado de código abierto y multiplataforma desarrollado por *JetBrains* que se utiliza para programar en *Python*. Proporciona un análisis de código, depuración gráfica, probador de unidad integrada y apoya el desarrollo web con el *framework Django* (JetBrains Pycharm, 2017).

1.5.7. Sistema Gestor de Base Datos

Un Sistema Gestor de Base de Datos (SGBD, en inglés *DBMS*: *Data Base Management System*) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. Es un sistema de software que permite la definición de bases de datos; así como la elección de las estructuras de datos necesarias para el almacenamiento y búsqueda de los datos, ya sea de forma interactiva o a través de un lenguaje de programación. Un SGBD relacional es un modelo de datos que facilita a los usuarios describir los datos que serán almacenados en la base de datos junto con un grupo de operaciones para manejarlos (Ramos, 2014).

Para garantizar la persistencia de los datos generados por el módulo, es empleado el SGBD *PostgreSQL* versión 9.4, el cual está orientada a objetos y libre, publicado bajo la licencia BSD. Este potente gestor implementa el estándar SQL92/SQL99 y soporta distintos tipos de datos, tipos fecha, monetarios, elementos gráficos y cadenas de bits. Permite la creación de tipos propios e incorpora una estructura de datos *array* (PostgreSQL Global Development Group, 2014).

1.5.8. Servidor Web

Servidor Web es un programa que gestiona cualquier aplicación en el lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación en el lado del cliente. El código recibido por el cliente suele

ser compilado y ejecutado por un Navegador Web. Para la transmisión de todos estos datos se utiliza algún protocolo. Generalmente se utiliza el protocolo *HTTP* para estas comunicaciones, perteneciente a la capa de aplicación de la arquitectura *TCP*.

Se empleará el servidor web *Apache* versión 2.2, el cual es una aplicación gratuita que convierte cualquier ordenador en un servidor web. Es de código abierto, flexible, rápido y eficiente. *Apache* permite negociar protocolos *HTTP* entre una máquina que haría de servidor web y los otros ordenadores que deseen ver un determinado sitio web. Es multiplataforma y gracias a su popularidad es fácil encontrar documentación para profundizar en las funcionalidades que ofrece. Se caracteriza por su gran escalabilidad, seguridad y rendimiento. Tiene soporte para varios lenguajes como *Perl*, *Python* y *PHP*, lo que permite desarrollar aplicaciones web de gran calidad (Apache Software Foundation, 2011).

MÓDULO DE APACHE PARA *PYTHON Web Server Gateway Interface* (*WSGI*)

WSGI es una interface simple y universal entre los servidores web y las aplicaciones web o *frameworks*. Es similar a la especificación *Java Servlet* o *ASP/ASP.NET*. En general, es mucho más simple que dichas especificaciones, y se basa en el estándar *CGI* con mejoras para hacerla reentrante y persistente (Python, 2017). Para la propuesta de solución se propone usar esta interfaz en su versión 4.5.10.

1.5.9. Herramientas para pruebas de software

Las pruebas de software, son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada (Pressman, 2010).

En el caso de la propuesta de solución de la investigación, este será validado a través de varias tipos y técnicas de pruebas, algunas de ellas de forma manual y otras mediante el uso de herramientas que permiten la realización de dicha tarea. Entre las herramientas existentes, se pretende utilizar las herramientas *Acunetix Web Vulnerability Scanner* y *Apache JMeter*, para las pruebas de seguridad y de carga y estrés, respectivamente.

Acunetix realiza automáticamente auditorías a aplicaciones web comprobando vulnerabilidades de Inyección *SQL*, *Cross site scripting* y otras vulnerabilidades que puedan ser explotadas por *hackers*.

Apache JMeter se caracterizar por ser libre de uso y mostrar los resultados de las pruebas en una amplia variedad de informes y gráficas, con gran cantidad de variables que permiten interpretar los resultados desde diferentes puntos de vista.

1.5.10. Metodologías de desarrollo de software

Como define Méndez (2010), una metodología de desarrollo de software es un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo producto de software. Puede seguir uno o varios modelos de ciclo de vida, es decir, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto.

***AUP*-UCI**

Para el desarrollo de la solución se emplea la variación de la metodología de Proceso Unificado Ágil (*AUP* por sus siglas en inglés), por ser la empleada como estándar para guiar el proceso productivo de la UCI.

La metodología *AUP* contribuye de manera significativa al desarrollo del software que se espera obtener, ya que esta metodología además de ser ágil es también adaptable a cualquier circunstancia que pueda surgir en el proceso de desarrollo del proyecto (Ambler, 2014). La UCI le ha realizado modificaciones con el fin de adaptarlo al ciclo de vida definido para la actividad productiva de dicha institución; de las 4 fases que encierra la metodología *AUP* se simplificaron a (Sánchez, 2015):

- *Inicio*: esta etapa se mantiene de *AUP*, pero se modifica el objetivo de la misma; realización de la planeación del proyecto en cuanto a un estudio de la organización del cliente, el alcance del proyecto, estimaciones de tiempo esfuerzo y costo que serán necesarios para decidir si se desarrolla el producto o no.
- *Ejecución*: en esta fase se recogen las actividades que desarrolla *AUP* de elaboración, construcción y transición. Se ejecutan las actividades requeridas para obtener el software y ajustar los planes del proyecto, realizando las tareas de modelado del negocio, obtención y análisis de los requisitos, elaboración de la arquitectura, diseño, implementación y liberación de producto.
- *Cierre*: en esta fase se analizan los resultados del proyecto, su ejecución y se llevan a cabo actividades formales de cierre del proyecto.

1.6. Conclusiones parciales del capítulo

Durante este capítulo se trataron los elementos teóricos que sustentan a la propuesta de solución del problema planteado, en este sentido se concluye:

- La sistematización de los conceptos fundamentales asociados al dominio de la presente

investigación y sus relaciones, proporcionó una mayor comprensión de la propuesta de solución.

- El análisis de las sintaxis de cada sentencia empleada para la manipulación de bases de datos relacionales, permitió establecer los elementos que deben tenerse en cuenta para la implementación de las mismas desde la propuesta de solución.
- El análisis de las herramientas existentes para el manejo de BDR, posibilitó identificar sus características y deficiencias que impiden sean empleadas como soporte tecnológico de apoyo a la asignatura SBD I en la UCI.
- El análisis de la documentación referente a la plataforma *RDB-Learning,* acerca de la metodología de desarrollo, las herramientas, tecnologías y lenguajes de programación empleados en la implementación de la misma, brindó la posibilidad de especificar el ambiente de desarrollo de la propuesta de solución.

CAPÍTULO 2 "Características y diseño del Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*"

2.1. Introducción

Establecer una estrecha relación entre el cliente y el equipo de trabajo en función lograr los objetivos, es una de las prioridades cuando se desea desarrollar software. El presente capítulo aborda los principales aspectos relacionados con las características de la propuesta de solución. Donde se define el modelo conceptual, según el objeto de estudio, haciendo uso de la metodología *AUP* en su variante para guiar el proceso productivo de la UCI. Se identifican los requisitos funcionales y no funcionales con los que debe cumplir la propuesta de solución, así como estilo arquitectónico y los patrones de diseño para lograr buenas prácticas en el diseño y posterior implementación de la solución propuesta. Igualmente se muestran los principales artefactos de ingeniería de software propuestos por la metodología *AUP*-UCI correspondientes a las funcionalidades.

2.2. Análisis

En esta fase se presentan las principales características asociadas a la propuesta de solución. Se realiza, además, un análisis de los principales conceptos asociados al dominio de la presente investigación evidenciado su relación por medio de un modelo conceptual.

2.2.1. Características de la propuesta de solución

Dadas las necesidades planteadas en la situación problemática de la presente investigación, la solución propuesta constituye un Módulo para la implementación de sentencias del lenguaje de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*.

Este módulo permite a los profesores registrados, crear ejercicios del tema de implementación de sentencias, los cuales podrán ser visualizados y luego resueltos por los estudiantes. Esta solución se alcanza desde la construcción grafica de las sentencias definidas en el en epígrafe 1.3.

Una vez creada la respuesta a un ejercicio determinado, la solución obtenida puede ser sometida a evaluación por el profesor que atiende el grupo (respuesta privada), como podrá ser enviada al foro al que pertenece el problema (respuesta pública) y se emitan comentarios de todos los usuarios del sistema. Una vez que el profesor revise la solución, este debe emitir su evaluación y enviarla al estudiante.

A lo largo del desarrollo del capítulo se describen los artefactos que permiten conceptualizar y comprender mejor el funcionamiento del módulo.

2.2.2. Modelo Conceptual

Un modelo conceptual es una representación visual en forma de diagrama de las clases conceptuales u objetos del mundo real que son significativos en un dominio de interés; no se trata de un conjunto de diagramas que describen clases u objetos de software con responsabilidades (Larman, 2004).

Con el objetivo de lograr un mejor entendimiento de los procesos que requieren automatización se realizó un modelo conceptual (ver Figura 10), con un total de nueve (9) clases y (9) relaciones, el cual describe los conceptos más importantes dentro del contexto del sistema, así como sus respectivas relaciones.

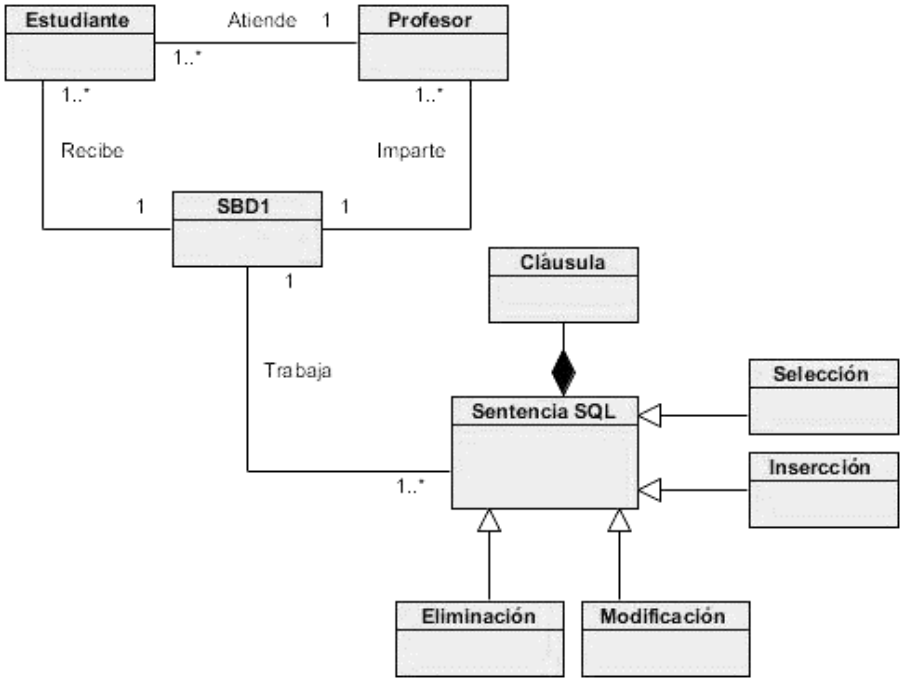


Figura 10: Modelo Conceptual

Como parte de este modelo, se aprecian un **Profesor** y un **Estudiante** como principales actores que intervienen en el proceso, donde un profesor puede atender a varios estudiantes. De igual manera un profesor imparte la asignatura **SBD1** que es recibida por varios estudiantes. En esta asignatura se trabajan con varios tipos de **Sentencias SQL.** Las mismas que están compuestas por **cláusulas**. Estas sentencias pueden ser de **Selección, Inserción, Modificación y Eliminación** de datos contenidos en una base de datos relacional.

2.3. Diseño

En esta fase se muestra los requisitos funcionales y no funcionales con lo que debe cumplir la propuesta de solución. Así como los artefactos generados a partir de la metodología seleccionada para guiar el proceso de desarrollo.

2.3.1. Especificación de requisitos

De acuerdo con lo planteado en Sommeville (2011), en la ingeniería del software, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto y establecen qué debe hacer el sistema, pero no cómo hacerlo. Son una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. De manera general estos requisitos son lo que el sistema debe hacer o una cualidad que el sistema debe poseer.

Requisitos funcionales

Luego de haber realizado un encuentro con el cliente y a través de una encuesta (ver Anexo 1) aplicada al mismo se obtuvo un total de veintiún (21) requisitos funcionales, los cuales se relacionan a continuación en la Tabla 2. La prioridad de cada requisito se establece en función de la importancia fijada por el cliente a partir de sus necesidades.

Tabla 2: Listado de requisitos funcionales

Código	Requisito Funcional	Prioridad
RF1	Crear ejercicio de implementación	Alta
RF2	Modificar ejercicio de implementación	Baja
RF3	Eliminar ejercicio de implementación	Media
RF4	Listar ejercicios de implementación	Media
RF5	Mostrar ejercicio de implementación	Alta
RF6	Crear un grupo docente	Alta
RF7	Modificar un grupo docente	Baja
RF8	Eliminar un grupo docente	Media

RF9	Listar grupos docentes	Media
RF10	Mostrar grupo docente	Media
RF11	Crear Foro para ejercicio de implementación	Alta
RF12	Modificar Foro para ejercicio de implementación	Baja
RF13	Eliminar Foro para ejercicio de implementación	Media
RF14	Listar Foros para ejercicio de implementación	Alta
RF15	Mostrar Foro con sus ejercicios de implementación	Alta
RF16	Crear una respuesta de implementación	Alta
RF17	Modificar una respuesta de implementación	Baja
RF18	Eliminar una respuesta de implementación	Media
RF19	Publicar respuesta de implementación en el foro	Alta
RF20	Enviar respuesta de implementación al profesor	Alta
RF21	Evaluar y emitir criterios de una respuesta de implementación privada	Alta

Requisitos no funcionales

Distribuidos, según el estándar de calidad internacional ISO/IEC 9126-1, en especificaciones de usabilidad, confiabilidad, portabilidad, eficiencia, funcionalidad y mantenibilidad, se obtuvo un total de dieciséis (16) requisitos no funcionales y quedan relacionados a continuación.

Usabilidad:

RnF 1: El módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-learning* deberá ser una aplicación web

RnF 2: El sistema deberá contar con una ayuda para los usuarios, la cual explicará en detalles

cómo usar el sistema y los conceptos relacionados con la implementación de sentencias de manipulación de BDR

RnF 3: La aplicación debe adaptarse a las estrategias marcarias de la UCI

RnF 4: La aplicación debe presentar una interfaz agradable e intuitiva para el usuario

Confiabilidad:

RnF 5: El sistema debe ser tolerante a fallos, y mostrar solo la información necesaria para orientar al usuario

Portabilidad:

RnF 6: La propuesta de desarrollo debe ser capaz de ejecutarse en los navegadores empleados en la UCI, así como adaptar su interfaz a cualquier dispositivo

Eficiencia:

RnF 7: El sistema debe permitir que los usuarios interactúen con él de manera concurrente

RnF 8: El tiempo de demora de una petición al servidor debe ser menor a cinco (5) segundos

Funcionalidad:

RnF 9: La aplicación debe gestionar y requerir información de usuarios para su uso **RnF 10:** La información manejada por el sistema estará protegida de accesos no autorizados

RnF 11: Ante los errores que puedan ocasionarse en el sistema no se deben mostrar detalles de información que puedan comprometer su seguridad e integridad

RnF 12: El sistema deberá ser utilizado por los usuarios con dominio uci.cu

RnF 13: El sistema deberá ser capaz, por cuestiones de seguridad, expirar la sesión del usuario una vez pasado diez (10) minutos de inactividad

Mantenibilidad:

RnF 14: Se debe hacer uso de los estándares de codificación definidos para la plataforma RDB-Learning

RnF 15: Se permitirá realizar modificaciones posteriores para adaptar mejoras al sistema o en caso que cambien las necesidades de los clientes

RnF 16: El software estará bien documentado de forma tal que el tiempo de mantenimiento sea

mínimo en caso de necesitarse

2.3.2. Historias de usuario

De acuerdo a lo que plantea Sánchez (2015), la metodología *AUP*-UCI, en su escenario 4 para la disciplina Requisitos, genera como uno de sus artefactos, las Historias de Usuario (HU), que consiste en una técnica para encapsular los requisitos del *software*, a través de un conjunto de tablas en las cuales el cliente describe brevemente las características que debe poseer el sistema.

Para el diseño de la propuesta de solución fueron generadas un total de veintiuna (21) Historias de Usuario, a continuación, se muestran dos (2) de estas, correspondientes a los RF-5 y RF-16 respectivamente, detallados en el sub-epígrafe 2.3.1.

Tabla 3: Historia de Usuario Mostrar un ejercicio de implementación

Historia de Usuario	
Número: HU_5	**Nombre**: Mostrar ejercicio de implementación
Programador responsable: Ortelio Francisco Hernández. Socarrás	**Iteración asignada**: 1
Prioridad: Alta	
Descripción: Permite que, a partir de una lista de ejercicios disponibles, se muestren los detalles del que sea seleccionado.	
Observaciones: Esta funcionalidad podrá se accedida por estudiantes y profesores. El contenido del ejercicio se mostrará en los detalles del mismo y en una ventana flotante desde el área de trabajo.	

Tabla 4: Historia de Usuario Crear una respuesta de implementación

Historia de Usuario	
Número: HU_16	**Nombre**: Crear una respuesta de implementación
Programador responsable: Ortelio Francisco Hernández. Socarrás	**Iteración asignada**: 1
Prioridad: Alta	

Descripción: Consiste en responder los ejercicios disponibles para la implementación de sentencias de lenguaje de manipulación de datos. La respuesta consistirá en una sentencia SQL construida desde un editor gráfico o de forma manual.
Observaciones: Tendrán acceso los estudiantes y profesores. Para la construcción de una sentencia *SELECT* debe tenerse en cuenta todas las cláusulas que la conforman (*WHERE*, *GROUP BY*, *HAVING*, *ORDER BY*, *LIMIT*). Además, se debe permitir el uso de las sentencias *INSERT*, *UPDATE* y *DELETE*.

2.3.3. Estilo arquitectónico

Modelo – Plantilla – Vista

El *framework* de desarrollo web *Django* emplea una modificación del estilo arquitectónico Modelo-Vista-Controlador (MVC), llamada *Model–Template–View* (MTV), que sería Modelo-Plantilla-Vista, está forma de trabajar permite que sea pragmático.

Como plantea Infante-Montero (2012), para comprender como funciona el MTV de *Django* se debe tener en cuenta, su analogía con el MVC de la siguiente manera:

- El modelo en *Django* continúa siendo Modelo
- La vista en *Django* pasa a llamarse Plantilla
- El controlador en *Django* pasa a llamarse Vista

A partir de lo planteado se asumirá el estilo arquitectónico MTV de *Django* del cual se explica su funcionamiento a continuación y se podrá observar en la Figura 11:

1- El navegador web envía una solicitud

2- La vista interactúa con el modelo para obtener los datos

3- La vista llama a la plantilla

4- La plantilla muestra la respuesta a la solicitud del navegador

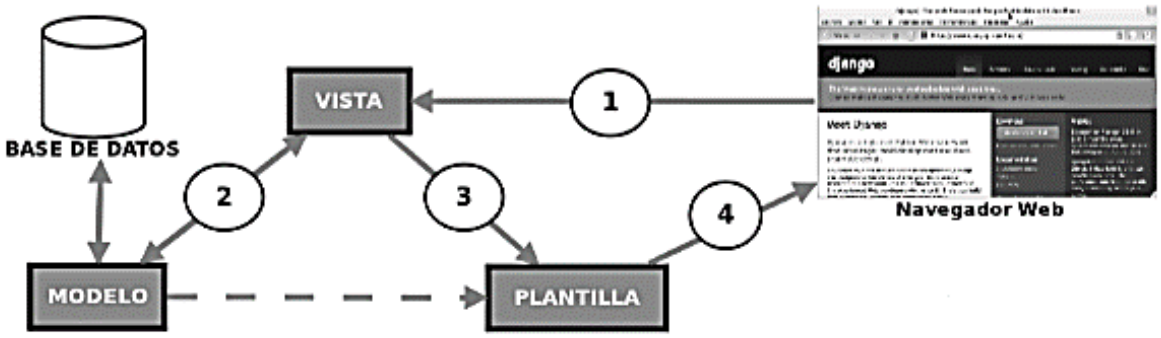


Figura 11: Funcionamiento del MTV de Django (Infante-Montero, 2012)

Del funcionamiento del MTV de *Django* se deriva lo siguiente (Infante-Montero, 2012):

- **El modelo:** Define los datos almacenados, es representado en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros igualmente posee métodos. Todo esto permite indicar y controlar el comportamiento de los datos.
- **La vista:** Su propósito es determinar qué datos serán visualizados, puede estar representado en forma de funciones o también en clases de *Python*. El *Object Relational Mapping* (*ORM*, es español Mapeo Relacional de Objetos) de *Django* permite escribir código *Python* en lugar *SQL* para hacer las consultas. Además, se encarga de tareas como: el envío de correo electrónico, autenticación con servicios externos, así como la validación de datos a través de formularios.
- **La plantilla:** Es la encargada de recibir los datos que provienen de las vistas y luego organizarlos para la presentación al navegador web. Es básicamente una página *HTML* con algunas etiquetas extras que son propias de *Django* que permiten que sea más flexible para los desarrolladores del *frontend*[9].

La imagen de la Figura 12 muestra la organización en carpetas de la propuesta de solución a partir de la utilización del *framework Django*.

[9] Es la parte del desarrollo web que se dedica de la parte frontal de un sitio web, en pocas palabras del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

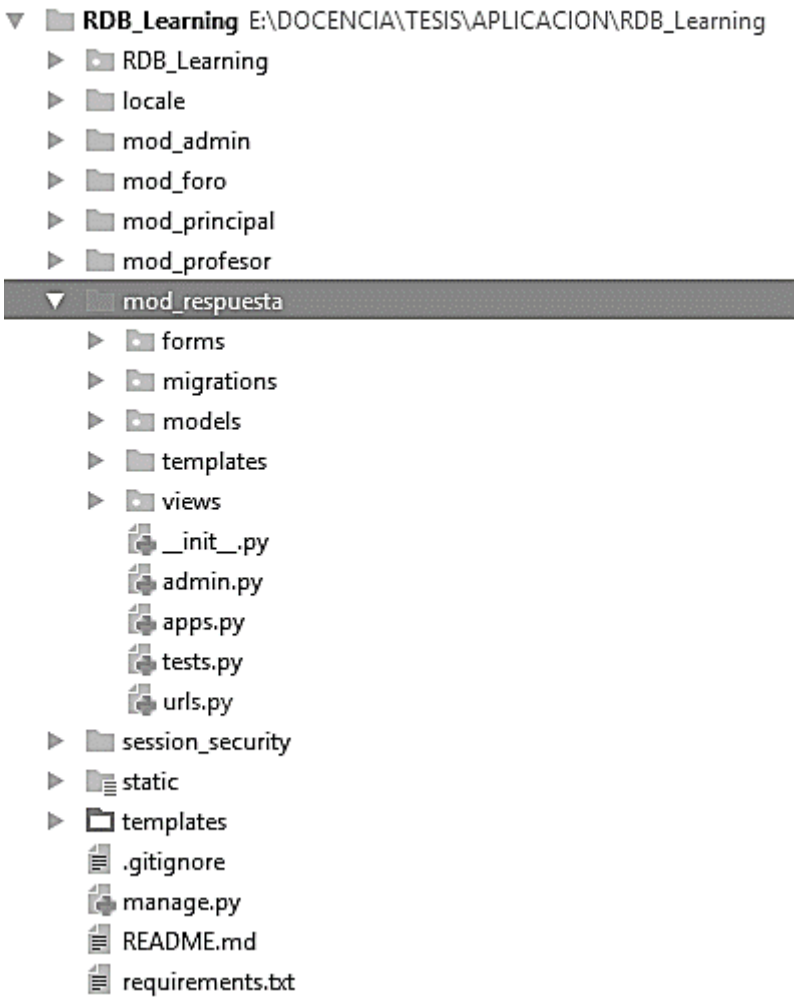


Figura 12: Estructura de la propuesta de solución

2.3.4. Diagrama de clases del diseño

Los diagramas de clase (DC) pueden usarse cuando se desarrolla un modelo de sistema orientado a objetos para mostrar las clases en un sistema y las asociaciones entre dichas clases (Sommerville, 2011).

Los diagramas de clases del diseño con estereotipos web, describen gráficamente las especificaciones del modelo, la vista y la plantilla de las historias de usuario descritas en el sub-epígrafe 2.3.2. Estas representaciones contienen información acerca de las clases, asociaciones, atributos, métodos y dependencias.

Para el diseño de la propuesta de solución fueron generadas un total de veintiún (21) DC, a continuación, se muestran dos (2) de estos correspondientes a los RF-5 y RF-16 respectivamente, detallados en el sub-epígrafe 2.3.1.

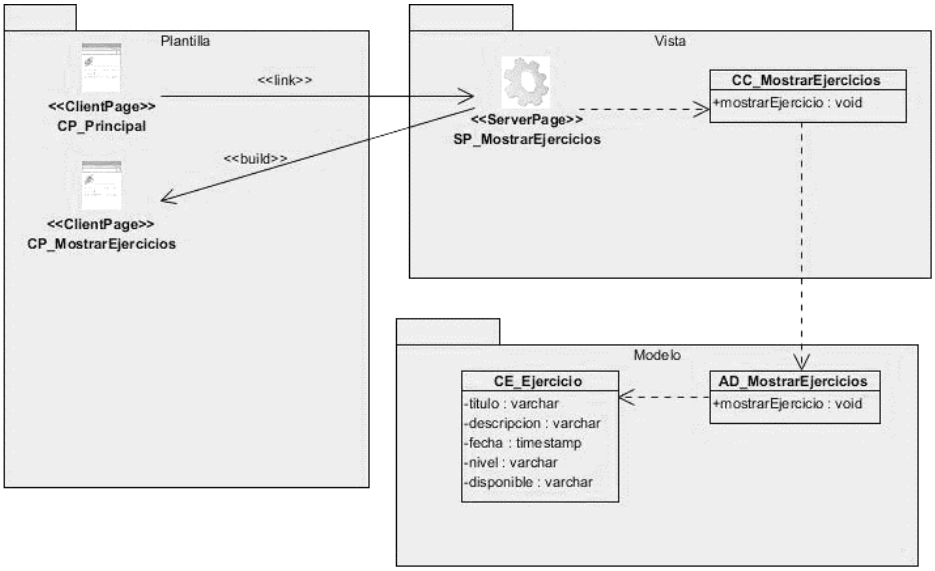


Figura 13: Diagrama de clases del diseño de la HU Mostrar ejercicio de implementación

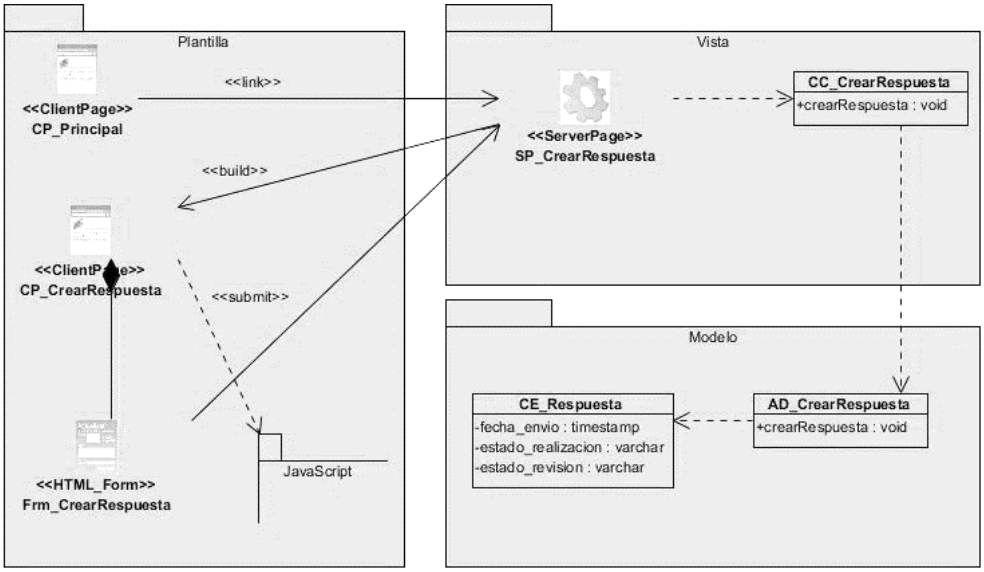


Figura 14: Diagrama de clases del diseño de la HU Crear respuesta de implementación

2.3.5. Patrones de diseño

De acuerdo a las valoraciones de Larman (2004), los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos, y presenta un esquema genérico demostrado con éxito para su solución; este último se especifica mediante la descripción de los

componentes que la constituyen, sus responsabilidades y desarrollos, así como también la forma como estos colaboran entre sí.

Seguidamente se presentan los patrones de diseño empleados en el desarrollo del módulo de implementación de sentencias de lenguajes de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*.

Patrones Generales de Software para la Asignación de Responsabilidades (GRASP):

Los patrones *GRASP* describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

GRASP es un acrónimo que significa *General Responsibility Asignment Software Patterns* (patrones generales de software para asignar responsabilidades). El nombre se eligió para indicar la importancia de captar (*grasping*) estos principios, si se quiere diseñar eficazmente el *software* orientado a objetos (Gamma, Helm, & R. Vlissides, 2009).

Según plantea Astudillo, Visconti y Hernán (2011), lo patrones *GRASP* son:

- **Experto:** El patrón garantiza que la responsabilidad de la creación de un objeto o la implementación de un método, recaiga sobre la clase que conoce toda la información necesaria para crearlo lo que contribuye a un adecuado encapsulamiento, favoreciendo la robustez y fácil mantenimiento del sistema.

  Las clases entidades **Respuesta** e **Implementacion**, son las expertas en la información relacionada con las respuestas a ejercicios de implementación.

- **Creador:** Se asigna la responsabilidad a una clase de crear cuando contiene, agrega, compone, almacena o usa otra clase, lo que brinda una alta posibilidad de reutilizar la clase creadora.

  Este patrón se pondrá de manifiesto en la implementación de la función **GuardarRespuesta** que se encargará de guardar las respuestas a un determinado ejercicio cuyo identificador coincide con uno pasado por parámetro, se crea una nueva instancia de la entidad **Respuesta**, para el ejercicio a resolver.

- **Bajo acoplamiento:** Es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases.

  Este patrón ya viene incluido con *Django* que permite un bajo acoplamiento entre las piezas, lo

que evita las dependencias, por ejemplo, a la hora de realizar cambios en las configuraciones de las *URL*[10], en la BD y las plantillas *HTML*, basta solo con realizarlo una sola vez.

- **Alta Cohesión:** Asigna responsabilidades de manera que una clase no tenga muchas funcionalidades no relacionadas o no realice un trabajo excesivo. Este patrón incrementa la claridad y facilita la comprensión del diseño.

  Una de las características de *Django* es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, se puede observar en el sistema que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

- **Controlador:** Asignar la responsabilidad de administrar un mensaje de evento del sistema a una clase que representa el sistema global, dispositivo, subsistema o representa un escenario de caso de uso en el que tiene lugar el evento del sistema.

  Este patrón es empleado en todo el sistema debido a que cada uno de los eventos generados por el usuario es redirigido a una clase o función controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión.

Patrones *Gang of Four* (*GoF*):

El catálogo de patrones más famoso es el contenido en el libro "*Design Patterns: Elements of Reusable Object-Oriented Software*", también conocido como: El libro *GOF* (*Gang-Of-Four Book*). Según este documento, estos patrones se clasifican según su propósito en creacionales, estructurales y de composición, mientras que respecto a su ámbito se clasifican en clases y objetos:

- ***Decorator* (Decorador):** Permite agregar responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades.

  En muchos lenguajes de programación, el patrón decorador se implementa a través de subclases (herencia). En *Python*, se puede emplear la función de decorador incorporado. Un decorador de *Python* es un cambio específico a la sintaxis de este lenguaje, que se utiliza para extender el comportamiento de una clase, método o función sin utilizar la herencia (Kasampalis, 2015).

[10] Del inglés *Uniform Resource Locator* (Localizador De Recursos Uniforme).

2.3.6. Modelo de datos

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad. Un esquema es un conjunto de representaciones lingüísticas o gráficas que describen la estructura de los datos de interés (Cuaran & Alonso-Guerrero, 2009).

El modelo de datos generado, como se observa en la Figura 15, está compuesto por dieciséis (16) relaciones y once (11) tablas, las cuales se explican a continuación, para un mejor entendimiento del modelo (de arriba abajo y de izquierda a derecha):

Perfil: almacena la información adicional referente a un usuario en el sistema, proporcionada por el sistema de identificación de la UCI. Se almacena avatar, credencial, carné de identidad, sexo, categoría, expediente, área, teléfono de la residencia, edificio y apartamento, así como municipio y provincia de procedencia. Contiene también la referencia al usuario al cual pertenece el perfil (*id_usuario*).

Usuario: registra, de un usuario del sistema, el atributo que lo identifica (*id_usuario*), su usuario uci, la contraseña, el correo, su(s) nombre(s) y apellidos, la fecha en que accedió por primera vez al sistema, así como la fecha de su última conexión.

Comentario: contiene los comentarios que son realizados en los foros, referentes a las respuestas de ejercicios publicadas en los mismos. Contiene además las referencias al usuario que realiza el comentario (*id_usuario*) y a la respuesta publicada que se comenta (*id_respuesta*).

Estudiante: esta entidad, es una especialización de la entidad usuario, por lo que hereda todos sus atributos. Del estudiante se desea conocer además su identificador (*id_estudiante*) y la referencia al grupo docente al que pertenece (*id_grupo_docente*).

Profesor: también se especializa de la entidad usuario. Se conoce de este el atributo que lo identifica (*id_profesor*).

Ejercicio: almacena la información referente a los ejercicios que son creados en el sistema, su identificador (*id_ejercicio*), el nombre, la descripción del ejercicio, su tipo, fecha de creación, fecha de modificación, los modelos *JSON* e imagen del ejercicio, si ha sido eliminado y si está disponible o no para ser resuelto. Se conoce además el profesor que lo crea (*id_profesor*) y el foro al que se asocia el

ejercicio (*id_foro*), atributo que puede ser nulo, si el ejercicio no se asocia a ningún foro.

Respuesta: de las respuestas se conoce su identificador (*id_respuesta*), el estado de finalización (terminado o pendiente), si ha sido revisada, el tipo de respuesta (privada o pública), la fecha de envío, la fecha de creación y fecha de la última modificación. Se almacena también, la referencia al ejercicio al cual pertenece la respuesta (*id_ejercicio*), al estudiante que la realizó (*id_estudiante*) y al profesor que la evalúa en caso de ser privada (*id_profesor*).

RespuestaRevision: Almacena los datos referentes a la revisión de las respuestas, y se conoce la respuesta que ha sido revisada, el profesor que la revisó, la evaluación emitida y el criterio del mismo. Así como también la fecha de la primera y de la última revisión.

Implementacion: esta entidad, hereda de la entidad respuesta, se conoce su identificador (*id_respuesta_implementacion*) y se almacena además la operación realizada en formato *JSON*, así como el código *SQL* generado.

GrupoDocente: contiene la información referente a los grupos docentes que son atendidos por los profesores y a los cuales pertenecen los estudiantes usuarios del sistema. Se conoce su identificador (*id_grupo_docente*), el nombre del grupo, el profesor que lo atiende (*id_profesor*), la fecha de creación, si ha sido eliminado y si está disponible o no.

Foro: almacena la información de los foros que son creados en el sistema, su identificador (*id_foro*), el tema con el que está relacionado, la descripción del foro, la fecha de creación, la forma de evaluación, el tipo de foro y el nivel de dificultad sobre el que estarán los ejercicios que se asocien a él. Se conoce además el profesor que crea el foro (*id_profesor*).

CAPÍTULO 2

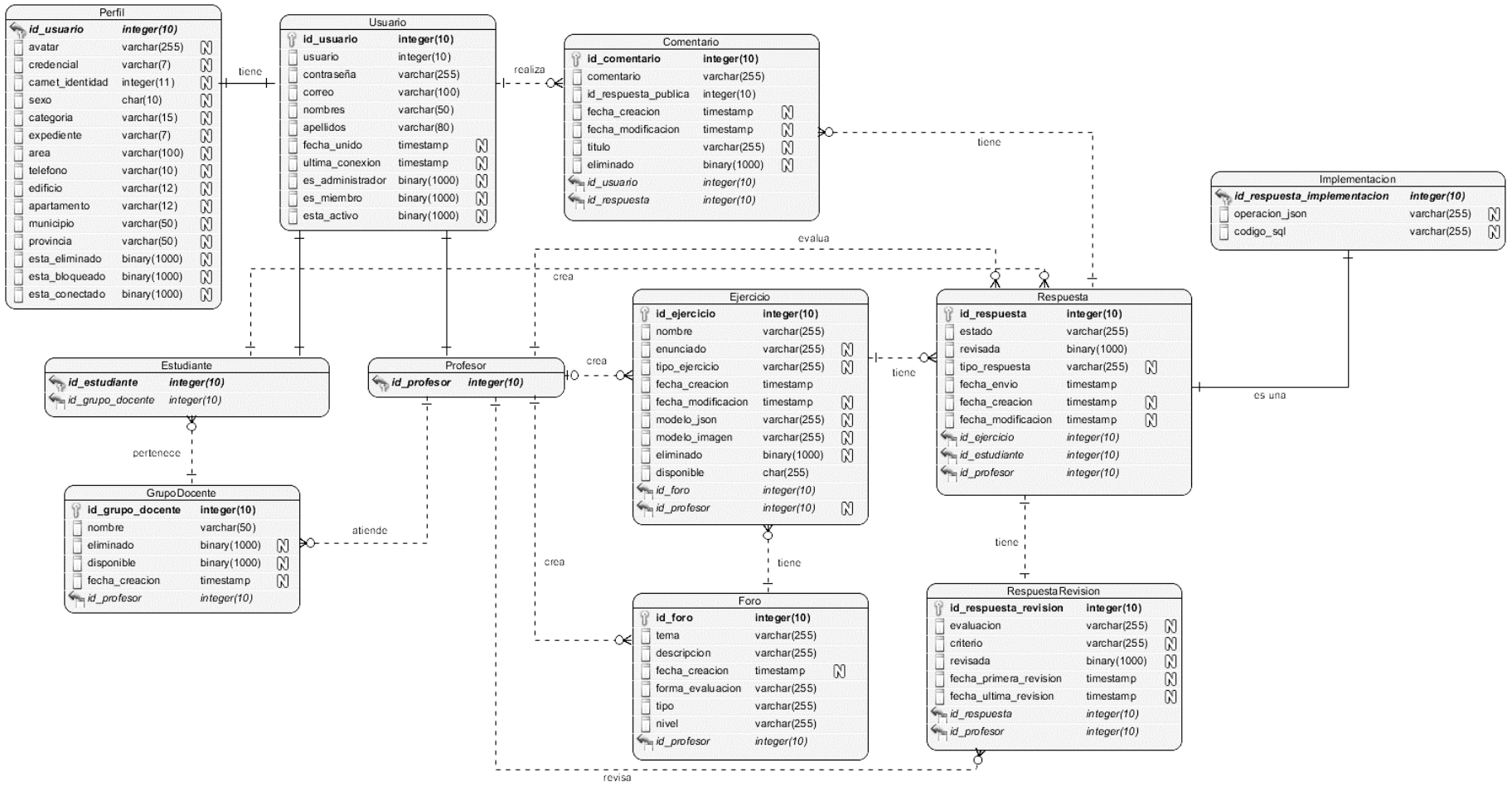


Figura 15: Modelo de datos

2.3.7. Modelo de despliegue

Un modelo de despliegue refleja la arquitectura en tiempo de ejecución de un sistema. En la Figura 9 se representan los dispositivos físicos de los artefactos de software en nodos como Servidor de Aplicaciones (*Apache*), Sistema Gestor de Base de Datos (*PostgreSQL*), Pasarela de autenticación del dominio mediante *WebServices* y el ordenador cliente, además especifica el protocolo y puerto que se emplea en la conexión entre cada nodo.

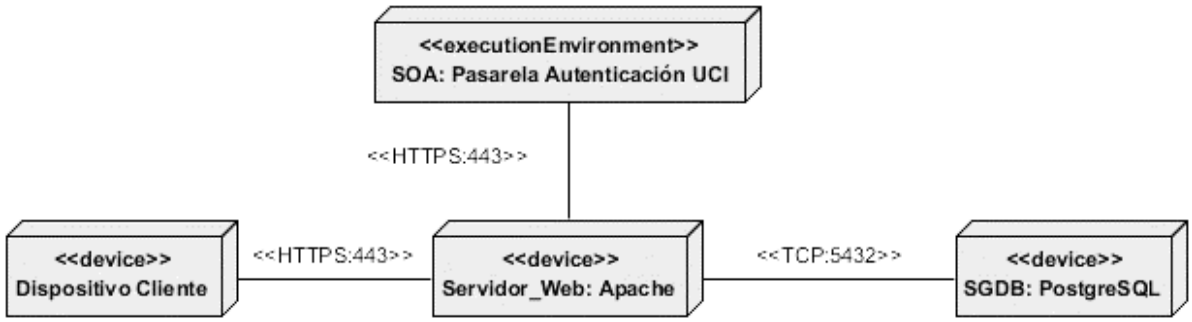


Figura 16: Modelo de despliegue de la solución propuesta

Este diagrama se considera para lograr un despliegue exitoso de la aplicación. En este se definen:

Dispositivo Cliente: Se refiere a las estaciones de trabajo (Cliente: PC, tableta, teléfono celular) que el usuario utilizará para conectarse, vía *HTTPS*, con el servidor de aplicaciones web.

Servidor de Bases de Datos: Almacena toda la información que brinda la Plataforma hospedada en el servidor de aplicaciones. La información es obtenida o modificada en dependencia del nivel de privilegio del usuario que realiza la petición. La comunicación con el servidor de aplicaciones es a través del protocolo *TCP* empleando el puerto 5432. Además, debe contar con una capacidad mínima de 500 MB de disco duro.

Pasarela Autenticación: Es la encargada de brindar la información de los usuarios de la UCI que se registran en la Plataforma hospedada en el servidor de aplicaciones. La comunicación la realiza a través del protocolo seguro *HTTPS*.

Servidor Web: Es el encargado de brindar la interfaz de la Plataforma para que los usuarios puedan hacer uso de esta, almacena todo el código fuente del sistema y se comunica por medio de los protocolos *TCP* con el servidor de bases de datos y por *HTTPS* con la pasarela de autenticación para usuarios de la *UCI*. Este debe disponer de un ordenador con sistema operativo *GNU/Linux* y cuyas propiedades mínimas sean un procesador *Intel Dual Core* con velocidad de 2.10 GHz, 2 GB de memoria *RAM*.

Además, se requiere de la instalación del módulo *WSGI,* así como las siguientes dependencias*: pyhton 2.7.11, python-suds 0.4, python-Pillow 2.7.0, python-psycopg 2 y python-django 1.9.5.*

2.4. Conclusiones parciales del capítulo

Luego de haber de realizar el análisis y diseño del módulo de implementación de sentencias de manipulación y haber generado los artefactos que dispone la metodología *AUP,* se puede concluye lo siguiente:

- El análisis de las características del sistema y el modelado del dominio, permitió identificar los principales requisitos funcionales y no funcionales del módulo de implementación de sentencias de manipulación de bases de datos relacionales, los que fueron agrupados y categorizados en sus respectivas historias de usuarios.
- La identificación de los patrones de diseño y el estilo arquitectónico de la solución propuesta, permite disminuir el impacto de los cambios futuros en el código fuente de la misma.
- El diseño de los diagramas de clases, facilitó el enfoque en cuanto a composición lógica y física de la propuesta de solución.
- La generación de todos los artefactos requeridos por el modelo de desarrollo, documentan la solución propuesta, lo cual facilita su posterior mantenimiento (actualización o adición de funcionalidades).

CAPÍTULO 3 "Implementación y pruebas del Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*"

3.1. Introducción

En el capítulo se aborda las actividades que se llevan a cabo durante la fase de implementación y pruebas. Se analiza el modelo de Implementación, así como la descripción de cómo los elementos del modelo de diseño se implementan en términos de componentes. Además, se diseña y aplican las pruebas para comprobar el correcto funcionamiento del módulo.

3.2. Modelo de implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Pressman, 2010).

3.2.1. Diagrama de componente

El diagrama de componentes muestra las relaciones estructurales entre los componentes de un sistema. Los componentes se consideran, unidades autónomas encapsuladas dentro de un sistema o subsistema que proporcionan una o más interfaces. Los diagramas de componentes son generalmente dirigidos al personal de aplicación de un sistema, presenta una comprensión temprana del sistema global que se está construyendo (BELL, 2004).

A continuación, se muestra el diagrama de componentes del Módulo implementación de sentencias de manipulación de bases de datos relacionales para la Plataforma *RDB-Learning*; cuya organización se encuentra acorde con el estilo arquitectónico MTV propuesto por el *framework Django* y descrita en el capítulo anterior de este trabajo. Donde los elementos del lado del cliente están almacenados en el paquete Plantilla.

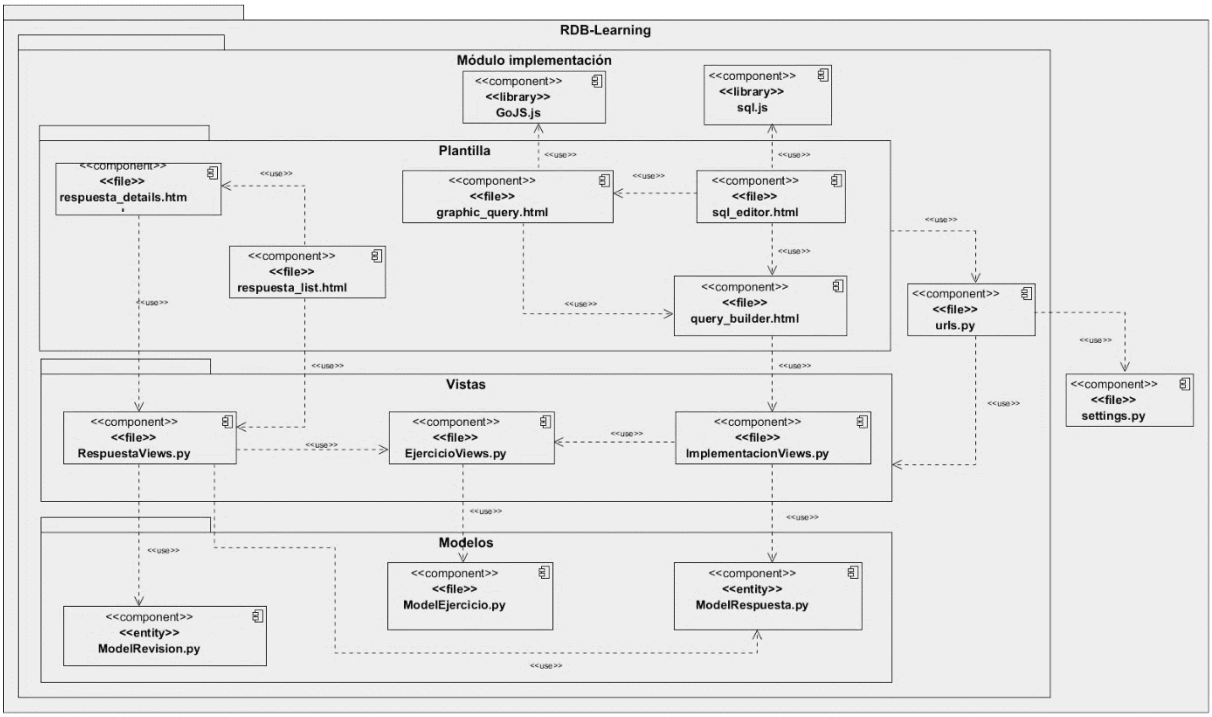


Figura 17: Diagrama de Componentes del Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales

A continuación, en la Tabla 5, se describen los elementos que conforman al diagrama de componentes de la Figura 17:

Tabla 5: Descripción de los componentes

	Componentes	**Descripción**
Plantillas	grafic_query.html	Este componente se encarga de diseñar gráficamente sentencias SQL de tipo *SELECT, UPDATE, INSERT, DELETE.* Emplea la librería *GoJS.js* para la construcción gráfica de las sentencias.
	sql_editor.html	Este componente muestra un editor de código *SQL* en el cual se autogenera la sentencia que se diseñe desde el *grafic_query.html*. Además permite, se le añadan otros elementos de forma manual y validar sintácticamente las sentencias haciendo uso de la librería *sql.js*.

	respuesta_list.html	Este componente muestra un listado con todas las respuestas enviadas al sistema permite filtrarla para ser mostrada tanto por ejercicio, por estudiante y las privadas (profesores), así como las públicas (foro del problema), o todas de manera general (solo usuario con permisos).
	Respuesta_details.html	Este componente muestra los detalles de una respuesta en que si esta es privada el profesor podrá evaluarla y enviar sus consideraciones al estudiante, en caso de ser publica se mostraría los comentarios hecho por todos los usuarios.
Vistas	ImplementacionViews.py	Este componente es el encargado de Crear y actualizar las respuestas de implementación.
	EjercicioViews.py	Este componente se encarga de gestionar los ejercicios en el sistema.
	RespuestaViews.py	Este componente se encarga de guardar (área personal), enviar al profesor (privada), publicar en el foro (pública) al que pertenece el ejercicio, evaluar la respuesta, eliminarla, mostrar los detalles de la misma, así como devolver un listado de estas.
Modelos	ModelEjercicio.py	Clase entidad que almacena los ejercicio disponibles en el sistema.
	ModelRespuesta.py	Clase entidad que almacena las respuestas de los ejercicios, enviadas al sistema.

	ModelRevision.py	Clase entidad que almacena los datos de las respuestas privadas tales como: la evaluación hecha por el profesor y su criterio.

3.3. Estándar de codificación

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del *software* y para obtener un buen rendimiento (Microsoft, 2017).

A continuación, se define el estándar de codificación basado en el empleado la plataforma *RDB-Learning*.

Tabla 6: Estándares de codificación a utilizar en la implementación del sistema

Tipo de estándar	**Descripción**
Organización del código	➢ El código en una página se organizará por bloques ➢ El formato del código se ajustará por la opción *Reformat Code* del *IDE* de desarrollo *PyCharm* ➢ La indentación se realizará solamente con tabulaciones, no debe utilizarse nunca los cuatro (4) espacios
Máxima longitud de las líneas	➢ Todas las líneas deben estar limitadas a un máximo de setenta y nueve (175) caracteres ➢ Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas ➢ En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas
Líneas en blanco	➢ Separar las funciones de alto nivel y definiciones de clases con dos (2) líneas en blanco

	➢ Las definiciones de métodos dentro de una clase deben separarse por una (1) línea en blanco ➢ Se puede utilizar líneas en blanco escasamente para separar secciones lógicas
Codificaciones	➢ Utilizar la codificación *UTF-8* ➢ Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”
Importación	➢ Las importaciones deben estar en líneas separadas ➢ Siempre deben colocarse al comienzo del archivo ➢ Deben quedar agrupadas de la siguiente forma: 1. Importaciones de la librería estándar 2. Importaciones terceras relacionadas 3. Importaciones locales de la aplicación / librerías ➢ Cada grupo de importaciones debe estar separado por una línea en blanco
Espacios en blanco en expresiones y sentencias	➢ Evitar utilizar espacios en blanco en las siguientes situaciones: • Inmediatamente dentro de paréntesis, corchetes y llaves • Inmediatamente antes de una coma, un punto y coma o dos puntos • Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función • Inmediatamente antes de un corchete que empieza una indexación • Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro ➢ Deben rodearse con exactamente un espacio los siguientes operadores binarios: • Asignación (=) • Asignación de aumentación (+=, -=, *=, /=) • Comparación (==, <, >, >=, <=, !=, <>, *in, not in, is, is not*) • Expresiones lógicas (*and*, *or*, *not*) ➢ Si se utilizan operadores con prioridad diferente se aconseja

	rodear con espacios a los operadores de menor prioridad ➢ No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto
Comentarios	➢ Los comentarios deben ser oraciones completas ➢ Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula ➢ Si un comentario es corto el punto final puede omitirse ➢ Los comentarios de una línea para aclaraciones del código aparecerán seguidos de los caracteres "//" en caso de código *JavaScript* mientras que en *Python* por el carácter "#" y deben ubicarse en la misma línea que se desea comentar ➢ Los comentarios de varias línea para organización del código aparecerán dentro de los caracteres "/** … **/" en caso de código *JavaScript*, mientras que en *Python* se hará con los caracteres '''…'''
Convenciones de nombramiento	➢ Nunca se deben utilizar como simple caracteres para nombres de variables los caracteres ele minúscula "l", o mayúscula "O", ele mayúscula "L" ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0) ➢ Los módulos deben tener un nombre corto y en minúscula. ➢ Los nombres de clases deben utilizar la convención "*CapWords*" (palabras que comienzan con mayúsculas) ➢ Los nombres de las funciones deben estar escrito en minúscula separando las palabras con un guión bajo "_" ➢ Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (_)

3.4. Pruebas de Software

El proceso de pruebas se centra en los procesos lógicos internos del *software*, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de

las pruebas para la detección de errores. Además, son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa (Pressman, 2010).

En este epígrafe se muestran los resultados de la estrategia de prueba diseñada para la propuesta de solución (ver Tabla 7), en función de garantizar y validar la calidad de este.

Tabla 7: Estrategia de pruebas

Tipo de prueba	Método (técnica) de prueba	Validación
Funcional	Casos de prueba (Caja Negra)	Valida las funcionalidades diseñadas para el sistema
Carga y estrés	*Software Apache JMeter*	Valida el comportamiento del sistema con distintos niveles de usuarios concurrentes y el consumo excesivo de sus recursos
Seguridad	*Software Acunetix*	Valida la confidencialidad, integridad y disponibilidad de los datos en el sistema
Usabilidad	Lista de Chequeo	Valida, a partir de sus características, la capacidad del sistema de cumplir con el propósito para el que fue diseñado
Aceptación	Carta de aceptación del cliente	Valida la funcionalidad del sistema a partir de la aceptación determinada por el usuario (cliente) del mismo

3.4.1. Pruebas funcionales

Las pruebas funcionales son aquellas que se llevan a cabo sobre la interfaz del software sin prestar atención al código, por lo que los casos de prueba son creados con el objetivo de demostrar que la entrada es aceptada de forma adecuada y que se produce una salida correcta. El diseño de estas pruebas se realiza con la intención de detectar funciones incorrectas o ausentes, errores en accesos a bases de datos externas, errores de interfaz, errores de rendimiento, y errores de inicialización y de terminación. Dentro de la prueba se incluyen la técnica de partición de equivalencia que será la empleada en la validación. Esta técnica divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba (Pressman, 2010).

Se realizaron los diseños de casos de prueba para validar la propuesta de solución implementada para comprobar que la herramienta agregue correctamente un ejercicio de implementación a la plataforma. A continuación, en la Tabla 8, se muestra las variables empleadas en el caso de prueba para el escenario “Crear ejercicio de Implementación”.

Tabla 8: Variables empleadas en el Caso de Prueba del RF1 Crear ejercicio de implementación

No.	Nombre del campo	Clasificación	Nulo	Descripción
1	Nombre	Campo de Texto	NO	Contiene solo letras
2	Enunciado	Área de Texto	NO	Contiene todo tipo de carácter.
3	Tipo	Campo de selección	SI	Contiene los tipos de ejercicios
4	Foro	Campo de selección	SI	Contiene los foro creados en el sistema.
5	Modelo JSON	Entrada de fichero	NO	Contiene un fichero JSON con la estructura del modelo de datos
6	Modelo Imagen	Entrada de fichero	SI	Fichero imagen del modelo de datos.
7	Publicado	Casilla de verificación	NO	Valores Verdadero o Falso

Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Tabla 9: Caso de Prueba del RF1 Crear ejercicio de implementación

Escenario	Descripción	V1	V2	V3	V4	V5	V6	V7	Respuesta	Flujo central
EC. 1.1 Crear ejercicio de implementación. Datos correctos	Se registran los datos de un ejercicio correctamente.	V	V	V	V	V	V	V	Muestra el mensaje "*Ejercicio registrado satisfactoriamente"*	El profesor rellena los campos de forma correcta y da clic en el botón Guardar.
		Título del ejercicio	Texto del enunciado (…)	Implementación	(Vacío)	Fichero JSON	Imagen del modelo	SI		
EC. 1.2 Crear ejercicio de implementación. Campos vacíos.	Existen campos obligatorios vacíos	V	I	V	V	I	I	V	Muestra el mensaje *"Existen campos obligatorios vacíos (campos)"*	El profesor llena los campos pero deja alguno vacíos y da clic en Guardar.
		Título del ejercicio	(Vacío)	Implementación	(Vacío)	(Vacío)	(Vacío)	NO		
EC. 1.3 Crear ejercicio de tipo implementación para un Foro	El tipo de ejercicio no coincide con el del Foro	V	V	V	I	V	V	V	Muestra el mensaje *"El tipo de foro no coincide con el tipo de ejercicio"*	El profesor llena los campos pero selecciona un foro que no es para ejercicios de implementación.
		Título del ejercicio	Texto del enunciado (…)	Implementación	Foro de tipo diseño	Fichero JSON	Imagen del modelo	NO		

CAPÍTULO 3

Resultados de las pruebas funcionales.

Las pruebas funcionales se realizaron en cuatro (4) iteraciones a medida que avanzaba el sistema, las cuales guiaron la calidad del mismo, y determinaron en cada momento si se estaba o no en condiciones de continuar avanzando con el ciclo de desarrollo.

Como resultado final de estas pruebas, se obtuvo, en una primera iteración, un total de quince (15) no conformidades, divididas en cinco (5) de ortografía, cuatro (4) de redacción, tres (3) de funcionalidad y tres (3) de validación. De estas, se resolvieron nueve (9), y seis (6) quedaron pendientes. En una segunda iteración, no se detectaron nuevas no conformidades y de las seis (6) pendientes, solo dos (2) se mantuvieron para la próxima iteración, donde fueron resueltas y no se detectan nuevas no conformidades. En una cuarta iteración no se identifican nuevas inconformidades, obteniendo, de esta manera, los resultados esperados. La gráfica de la Figura 18, muestra los resultados descritos anteriormente.

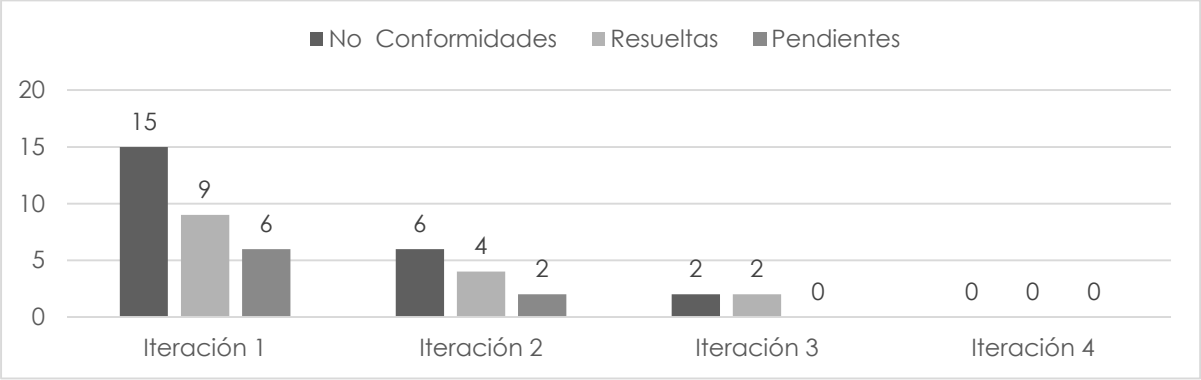


Figura 18: Resultados de las pruebas funcionales

Las no conformidades de funcionalidad, estuvieron relacionadas con crear un ejercicio, evaluar una respuesta y acceder a la ayuda. En el primer caso, el sistema no permitía crear el ejercicio sin asociarlo a un foro, sin embargo, el problema fue resuelto al permitir que el atributo *id_foro* de la entidad *Ejercicio*, el cual estaba especificado como campo obligatorio, pudiese tomar valor nulo. En el segundo caso, al calificar una respuesta solo se modificaba la evaluación de la primera que se encontraba en la lista de las respuestas para evaluar. El problema estaba dado porque el identificador de la ventana emergente de evaluación se especificó como atributo único, por lo que se le adicionó el identificador de la respuesta que se pretendía evaluar en cada caso para dar solución a este problema. En el último caso, se intentaba acceder a la ayuda teórica del módulo, pero dicho botón no accedía a la página correspondiente a la ayuda esto se debía a que la *URL* se encontraba mal configurada por lo que se corrigió la *URL*.

Las no conformidades de validación fueron detectadas en el formulario para crear un ejercicio, donde este permitía introducir caracteres extraños en el nombre de un ejercicio, al igual que crear un ejercicio de implementación y asociarlo a un foro para otro tipo de ejercicios, así como crear un ejercicio de este tipo con un fichero *JSON* inválido. La solución para los caracteres extraños fue validar el nombre de un ejercicio mediante una expresión regular que permite solo el uso de caracteres alfa-numéricos, las tildes, las diéresis, espacios, puntos y el guion bajo. Mientras que, para crear un ejercicio asociado correctamente a un foro, se validó que el tipo de ejercicio fuese el mismo que el del foro, lo cual no permite una asociación incorrecta. El sistema muestra un mensaje de error si se intenta realizar esta acción. Para el último caso, el de crear un ejercicio subiendo un fichero *JSON* incorrecto este fue resuelto realizando una revisión de la estructura para comprobar que sea la adecuada. De igual manera el sistema responde con un mensaje en caso de no cumplir con dicha estructura.

3.4.2. Pruebas de carga y estrés

La prueba de carga y estrés se refiere, generalmente, a la práctica de comprobar el comportamiento de una aplicación mediante cargas o entradas pesadas. Las mismas se realizan con el fin de verificar si el sistema satisface los requisitos de rendimiento para situaciones críticas como pueden ser: la cantidad límite de usuarios accediendo de forma concurrente a los servicios brindados, documentos extremadamente grandes, cantidad de transacciones que se pueden procesar de forma concurrente cada minuto, tiempo de respuesta, entre otros (ITI, 2017).

Para realizar las pruebas de carga y estrés se efectúo un análisis de la cantidad de usuarios en la Universidad. Del cual se obtuvo que la misma cuenta con cerca de 8000 usuarios, entre estos, alrededor de 2700 son estudiantes. Teniendo en cuenta que el módulo está pensado para el apoyo de la asignatura SBD I, se tuvo en cuenta la matrícula del 2do año que es de cerca de 700, de los que se toma muestras para casos críticos en que se conecten entre 100 y 200 usuarios simultáneamente.

Las pruebas se desarrollaron con el apoyo de la herramienta *Apache JMeter 2.12*, en la que se simuló el entorno donde debe interactuar el modulo para obtener la información más correcta acerca del comportamiento y resultados en general. Por lo que fue elegido un ambiente con las siguientes características:

Hardware:

- Microprocesador: *Intel Core*(TM) i3 – 3110M *CPU* @ 2.40 GHz
- Memoria RAM: 4GB

- Tarjeta de Red: *Ethernet* 10/100 Mbps

Software:

- Sistema Operativo: *GNU\Linux Ubuntu* 16.04 Arquitectura de 64bits

A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas al módulo:

Muestra: Cantidad de peticiones realizadas para cada *URL*.

Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.

Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.

Min: Tiempo mínimo que demora un hilo en acceder a una página.

Max: Tiempo máximo que demora un hilo en acceder a una página.

Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.

% Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.

Rendimiento (Rend): El rendimiento se mide en cantidad de solicitudes por segundo.

Kb/s: Velocidad de carga de las páginas.

Como se muestra en la Tabla 10, se simularon las peticiones realizadas al módulo por un total de 100, 500 y 1000 usuarios simultáneamente, realizando hasta 5 peticiones por segundo, obteniéndose los siguientes resultados:

Tabla 10: Resultado de las pruebas de carga y estrés

Usuarios	Muestras	Media	Mediana	Min	Max	Línea 90%	% Error	Rend.	Kb/s
100	500	1642	1125	91	4571	2819	0.00%	45.56	194.8
500	2500	1285	1078	125	4946	2096	0.45%	216.3	542.3
1000	5000	1225	1023	69	4761	1703	2.89%	437.7	974.6

Las pruebas realizadas muestran que el módulo es capaz de responder a 500 peticiones de 100 usuarios conectados simultáneamente en un tiempo promedio de 1642 milisegundos (1.6 segundos aproximadamente) con 0 % de error, esto evidencia que el módulo puede procesar la carga esperada para esta cantidad de usuarios.

Por otra parte, se realizaron 2500 peticiones iniciadas por 500 usuarios y en este caso el módulo

respondió en 1285 milisegundos (1.3 segundos aproximadamente) como tiempo promedio. Esto demuestra que el módulo es capaz de procesar la carga esperada para este número de usuarios, aunque fallo en un 0.45% de las peticiones.

Por último, y con el objetivo de analizar el comportamiento del módulo en condiciones extremas, se realizó una prueba de estrés para un total de 1000 usuarios conectados simultáneamente. En este caso, el módulo responde a las 5000 peticiones en un tiempo promedio de 1225 milisegundos (1.2 segundos aproximadamente), pero falla el 2.89% de las peticiones.

3.4.3. Pruebas de seguridad

Las pruebas de seguridad permiten realizar una evaluación de los sistemas desde el punto de vista externo y sin conocimiento previo de este. Tienen como objetivo hacer un análisis con el fin de encontrar fallos de seguridad tanto en el diseño como en la implementación de la aplicación. Además, buscan medir la confidencialidad, integridad y disponibilidad de los datos, partiendo de la identificación de amenazas y riesgos en el uso de interfaces de usuarios final. Una vez terminadas las pruebas es posible medir y cuantificar los riesgos a los cuales se ven expuestos aplicativos de la infraestructura interna y externa.

Resultados de las pruebas de seguridad.

Las pruebas de seguridad se aplicaron con ayuda de la herramienta *Acunetix Web Vulnerability Scanner* 9.5 que establece alertas de tipo: alta, media, baja e informacional, realizándose en dos iteraciones del desarrollo de la propuesta solución

En una primera iteración se obtuvo un total de 15 alertas de seguridad, de las cuales 6 clasifican de nivel medio, 6 de nivel bajo y 3 informativas. De las de nivel medio, destacó el uso de protocolo no seguro para el envío de datos, así como los mensajes de error que se muestra en el modo DEBUG de *Django* para el desarrollo.

Para las de nivel bajo, se detectó problemas para la protección de contra ataques de fuerza bruta a la página de autenticación, así como directorios que pueden ser accesibles directamente sin pasar la autenticación y la protección de las cookies y las sesiones en el navegador. De carácter informativo fue detectadas una dirección de correo y una posible cuenta de usuario en un fichero (ver Figura 19).

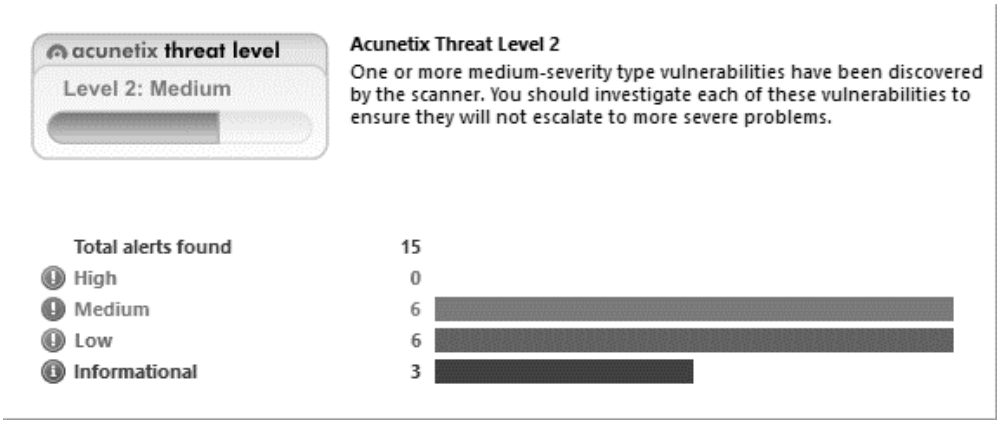


Figura 19: Pruebas de seguridad, primera iteración (Acunetix WVS)

En la imagen de la Figura 20, se puede apreciar el resultado final luego de corregir todas las alertas detectadas por la herramienta.

Figura 20: Pruebas de seguridad. Resultado final de la segunda iteración (Acunetix WVS)

3.4.4. Pruebas de usabilidad

En el contexto del desarrollo de software, la usabilidad está considerada como uno de los factores de calidad de mayor importancia para el éxito de un proyecto. De manera general, el término usabilidad es empleado para referirse a la capacidad que posee un producto de ser utilizado por los usuarios de forma fácil, eficiente y con satisfacción, en un determinado contexto de uso. La usabilidad se describe como un término multidimensional en el que intervienen cinco atributos fundamentales: capacidad de aprendizaje, eficiencia en el uso, facilidad de memorizar, tolerante a errores y subjetivamente satisfactorio (Nielsen, 2010).

La realización de pruebas de usabilidad contribuye, en cierta medida, a la adquisición de aplicaciones de alta calidad y gran facilidad de uso por parte de los usuarios finales (Alonso, Y. H.; Fortún, L. L., 2014).

Para ello, especialistas del Departamento de Evaluación de Productos de Software han establecido una lista de chequeo que proporciona un conjunto de preguntas clasificadas en nueve (9) categorías.

Resultados de las pruebas de Usabilidad

A continuación, la Tabla 11 muestra los resultados de la aplicación de la lista de chequeo al módulo de implementación de sentencias de manipulación de bases de datos relacionales para la Plataforma *RDB-Learning*.

Tabla 11: Resultados de las pruebas de Usabilidad empleando lista de chequeo

Categoría de indicadores	**Indicadores**	**Evaluados**	**Correctos**
Visibilidad del sistema	17	14	13
Lenguaje común entre sistema y usuario	11	7	7
Libertad y control por parte del usuario	29	14	14
Consistencia y estándares	33	22	21
Estética y diseño minimalista	18	9	9
Prevención de errores	8	6	6
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	7	1
Ayuda y documentación	11	4	2
Flexibilidad y eficiencia de uso	6	5	4
TOTAL	**144**	**88**	**77**

En la tabla anterior se puede apreciar que, de un total de 144 indicadores de usabilidad, fueron evaluados 88, de los cuales el módulo de implementación de sentencias de manipulación de bases de datos relacionales, cumple con 77 indicadores, cifra que representa el 87.5%. Para una mejor comprensión de los resultados obtenidos se han representado los mismos en la gráfica de la Figura 21.

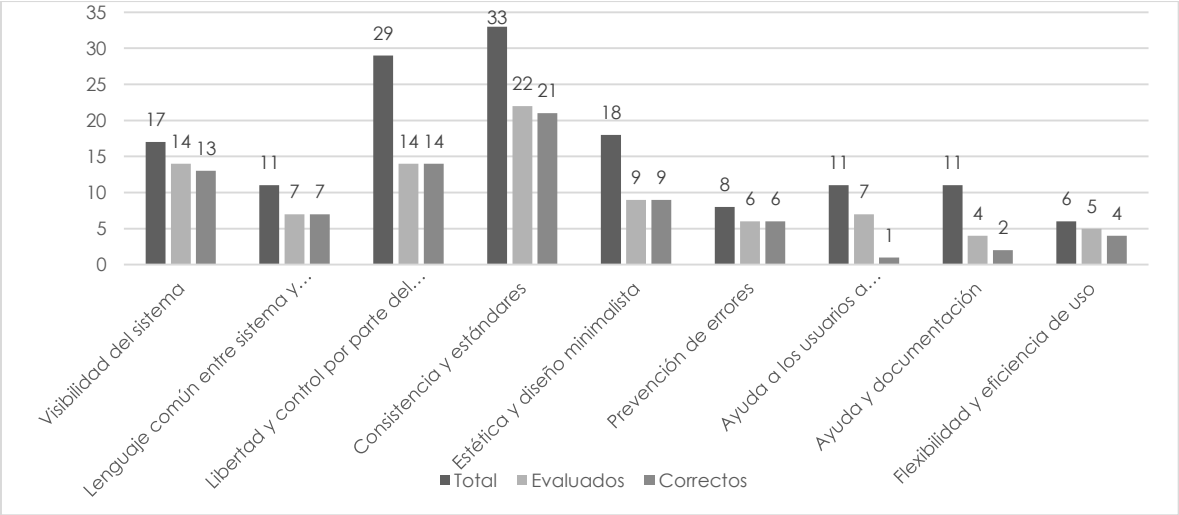


Figura 21: Cumplimiento de indicadores de la lista de chequeo de usabilidad aplicada

Después de analizar los resultados obtenidos en las pruebas de usabilidad, se identificaron nueve (9) indicadores con posibles mejoras de acuerdo al alcance del presente trabajo. Estos fueron implementados para facilitar el uso y la interacción entre el usuario y las operaciones que se realizan en el módulo; incrementando el nivel de usabilidad a 95.45 %. En la gráfica de la Figura 22 se representa el estado del nivel de usabilidad resultante después de terminada la prueba.

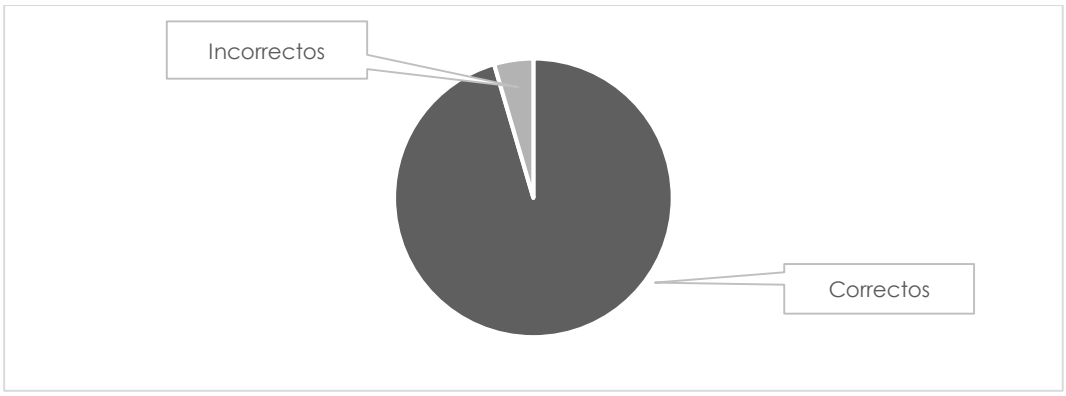


Figura 22: Nivel de usabilidad resultante

3.4.5. Pruebas de aceptación

Para realizar la prueba de aceptación, se le entregó la aplicación al cliente, el cual emitió su criterio a través de una carta de aceptación, a partir de sus consideraciones respecto a las ventajas que ofrece el módulo y las necesidades que resuelve. Para respaldar dicho criterio, se solicitó la colaboración de un grupo de expertos de la asignatura SBD I. Como instrumento fue utilizado una encuesta (ver Anexo 2) a través de la cual se puede validar el nivel de aceptación del módulo de implementación de sentencias

de manipulación de bases de datos relacionales en los usuarios finales. La encuesta fue realizada a cuatro (4) profesores de SBD I de la Facultad 1, los cuales cuentan con un promedio de 4 años de experiencia impartiendo la asignatura.

La tabla 12 resume el resultado de los juicios emitidos por los encuestados.

Tabla 12: Resultado de las encuestas realizadas para la Pruebas de Aceptación

Parámetros evaluados	**Niveles de valoración**									
	MA		**A**		**PA**		**NA**		**Total**	
	N	**%**	**N**	**%**	**N**	**%**	**N**	**%**	**N**	**%**
1	4	100	0	0	0	0	0	0	4	100
2	2	50	2	50	0	0	0	0	4	100
3	2	50	2	50	0	0	0	0	4	100
4	3	75	1	25	0	0	0	0	4	100
5	4	100	0	0	0	0	0	0	4	100

Para el procesamiento y análisis de la información obtenida se analizaron las respuestas de cada uno de los parámetros que aparecen en la encuesta. De esta forma se presentan los resultados teniendo en cuenta que los niveles empleados para la valoración fueron: **MA**: Muy adecuado, **A**: Adecuado, **PA**: Poco adecuado y **NA**: No adecuado. En la tabla, **N** se refiere a la cantidad de encuestados que emitieron una valoración determinada y % al porciento que representa con respecto al total de encuestados.

Los resultados antes mostrados, son representados de forma gráfica a continuación, en la Figura 23:

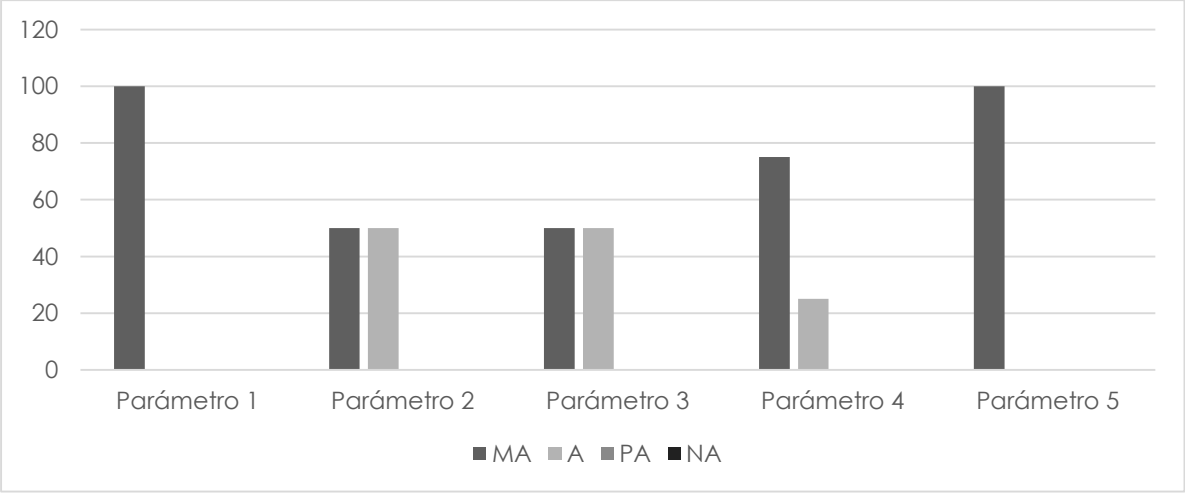


Figura 23: Resultados de la encuesta de aceptación

Como se observa, al analizar los resultados, se pudo comprobar que el 100% de los encuestados

consideran muy adecuado, el tratamiento de los aspectos teóricos de la asignatura SBD I en el módulo de implementación de sentencias de manipulación de BDR. El nivel de apoyo que brinda al profesor el uso del módulo para la orientación y evaluación de ejercicios. La contribución al tema de implementar sentencias de manipulación de bases de datos relacionales en la asignatura, fue valorado de muy adecuado por el 50% de los encuestados. El otro 50% considera que son adecuados. Respecto a la presentación de una interfaz agradable e intuitiva para el usuario, el 75% de los encuestados considera que es muy adecuada, por otra parte, el 100% considera muy adecuada el nivel de usabilidad del mismo.

Teniendo en cuenta los criterios y otras consideraciones de los profesores encuestados, se evidencia que la solución implementada tiene un nivel satisfactorio de aceptación entre ellos. Los aspectos evaluados, que están en concordancia con el objetivo general de la investigación, fueron valorados todos entre los niveles de adecuado y muy adecuado. Esto demuestra el correcto cumplimiento del mismo, desde el punto de vista de los expertos. Además, se obtuvo un conjunto de recomendaciones y valoraciones que aportan mejoras a la propuesta de solución, las cuales se tiene en cuenta para el resultado final de la investigación, así como futuras profundizaciones sobre la misma.

3.5. Interfaces principales

Una vez finalizado el desarrollado del software es posible visualizar las pantallas principales del módulo de implementación de sentencias de manipulación de BDR, donde se observa el resultado obtenido durante la implementación de las historias de usuarios descritas en el capítulo anterior.

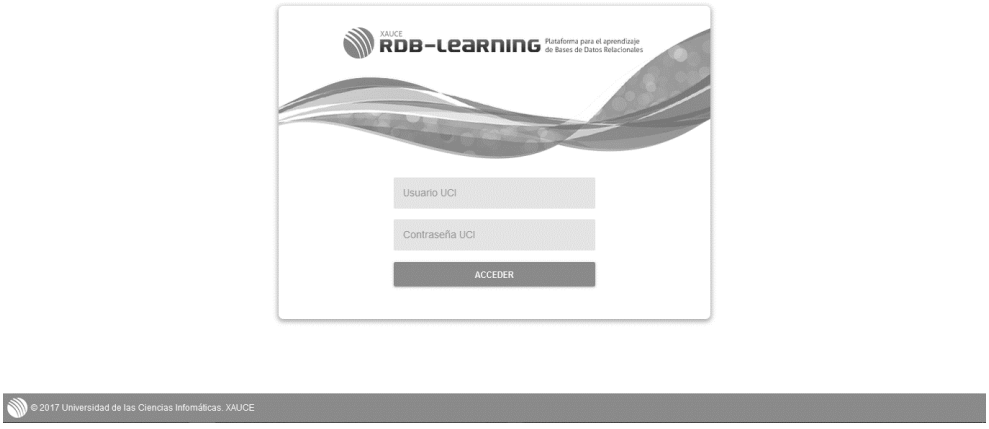


Figura 24: Interfaz de acceso al sistema

Figura 25: Interfaz Página principal

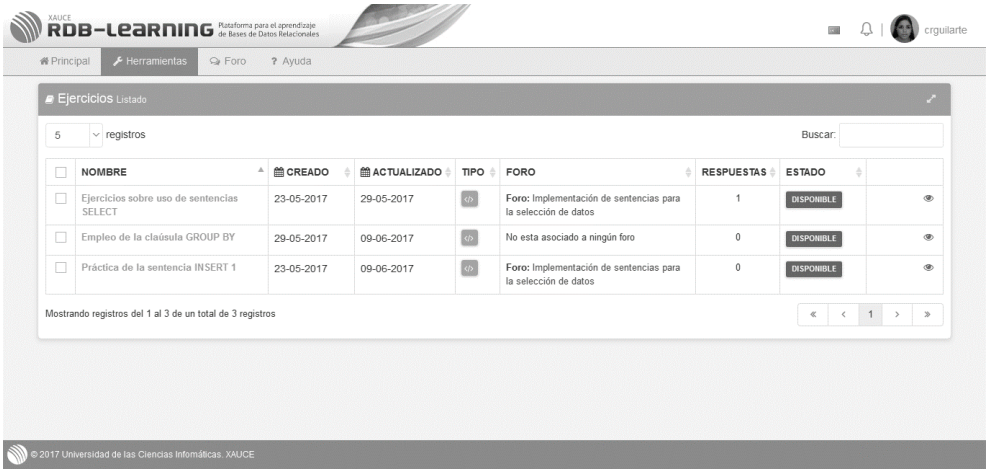


Figura 26: Interfaz Lista de ejercicios de implementación

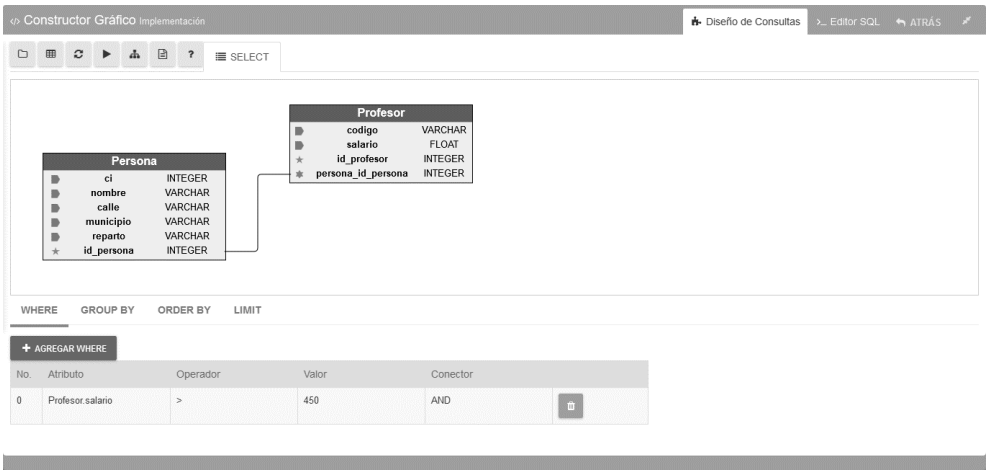


Figura 27: Interfaz Área de trabajo de implementación

3.6. Conclusiones parciales del capitulo

En este capítulo se abordaron los elementos de la implementación del módulo de implementación de sentencias de manipulación de BDR, así como las pruebas realizadas al mismo y los resultados obtenidos; lo que permite concluir:

- La confección del diagrama de componentes permitió una mejor comprensión de la estructura de los componentes de la propuesta de solución.
- El empleo de un estándar de codificación permitió garantizar la alta calidad, minimización errores y mayor limpieza en el código fuente. Lo que logra así pueda ser mantenido fácilmente y reutilizado por otros desarrolladores que lo necesiten.
- La implementación de la propuesta de solución facilitó la obtención de una aplicación funcional lista para su uso.
- La validación de la propuesta de solución, mediante una estrategia de pruebas de software, resulto que la misma cumple con los requisitos definidos por el cliente.

CONCLUSIONES

CONCLUCIONES GENERALES

El presente Trabajo de Diploma concluye con el desarrollo de un Módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales para la Plataforma *RDB-Learning*, el cual sirve de apoyo al proceso enseñanza-aprendizaje de la asignatura SBD I y contribuye a la implementación de sentencias de manipulación de bases de datos relacionales en dicha asignatura. Esto se debe a que el módulo fue desarrollado como tecnología del aprendizaje y el conocimiento que provee al estudiante una herramienta para facilitar la práctica en las actividades de auto-preparación y al profesor, para guiar, controlar y evaluar estas actividades.

Se destaca, además:

- El análisis y la fundamentación teórica de los principales conceptos asociados a la investigación, permitió comprender el alcance de la misma.
- El análisis de las herramientas informáticas existentes para la implementación de sentencias de manipulación de bases de datos relacionales y el estudio de la documentación de la plataforma *RDB-Learning*, permitió definir el ambiente de desarrollo para la implementación de la propuesta de solución.
- La combinación de conocimientos adquiridos de las distintas áreas del conocimiento como son la programación, ingeniería y gestión de software, bases de datos, entre otras posibilitó el análisis, diseño e implementación de la propuesta de solución.
- La validación a través de la definición de una estrategia de prueba, permitió comprobar el funcionamiento correcto del módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*, a partir del cumplimiento de los requisitos definidos por el cliente.

RECOMENDACIONES

Con el cumplimiento del marcado objetivo en la presente investigación y luego de alcanzado el resultado esperado, se proponen las siguientes recomendaciones:

- Incorporar al módulo la capacidad de autocorrección automática de los errores en las sentencias mientras se están diseñando
- Implementar un analizador semántico al editor de código *SQL*

REFERENCIAS

Academic. (2012). Obtenido de Enciclopedia Universal: http://enciclopedia_universal.esacademic.com/19671/Modelado_del_Software

Alonso, Y. H.; Fortún, L. L. (2014). Indicadores para evaluar la usabilidad en aplicaciones web. *1ra Conferencia Científica Internacional UCIENCIA 2014* (pág. 10). La Habana, Cuba: Universidad de las Ciencias Informáticas.

Ambler, S. (10 de Noviembre de 2014). *Ambysoft.* Recuperado el 20 de enero de 2017, de Ambysoft: http://www.ambysoft.com/unifiedprocess/agileUP.html

Apache Software Foundation. (20 de mayo de 2011). *The Apache HTTP Server Project.* Recuperado el 6 de diciembre de 2016, de http://httpd.apache.org/

Astudillo, M., Visconti, & Hernán. (2011). *Fundamentos de Ingeniería de Software.* Universidad Técnica Federico Santa María. Recuperado el 28 de febrero de 2017

Ávila Barrientos, E. (2014). *Formación de usuarios de la información mediante aplicaciones Web.* doi:DOI 10.5195/biblios.2014

BELL, D. (2004). *UML basics.* Recuperado el 25 de marzo de 2017, de The component diagram: https://www.ibm.com/developerworks/rational/library/dec04/bell/

Cabero-Almenara, J. (2004). La utilización de las TIC, nuevos retos para las universidades. *17(No. 3 Especial)*.

Cuaran, J. A., & Alonso-Guerrero, J. (Noviembre de 2009). DISEÑO DE PROYECTOS CON SIG. *SIG Aplicado a la Consulta Catastral de los Predios Urbanos del Municipio de Sevilla*. Sevilla, España: UNIVERSIDAD DEL VALLE.

Delgado, J. (28 de junio de 2011). *Teoría de Programación*. Recuperado el 2 de diciembre de 2016, de http://teoria-de-programacion.globered.com/categoria.asp?idcat=39

Django. (2016). *The Django Book*. Recuperado el 5 de diciembre de 2016, de http://www.djangobook.com/

e-ABC. (2017). *e-ABC*. Recuperado el 20 de mayo de 2017, de Definición de e-Learning: http://www.e-abclearning.com/definicione-learning

e-ABC. (2017). *e-ABC*. Recuperado el 20 de mayo de 2017, de Definición de e-Learning: http://www.e-abclearning.com/queesunaplataformadeelearning

EMS Database Management Solutions, Inc. (2017). *SQL Manager* . Obtenido de http://sqlmanager.net/en/products

REFENCIAS BIBLIOGRÁFICAS

Euphoria IT, L. (2008). *euphoriait.* Recuperado el 5 de diciembre de 2016, de http://euphoriait.com

Fernandez, N. (24 de abril de 2013). *Visual SQL Builder. Construye Visualmente Sentencias SQL.* Recuperado el 3 de diciembre de 2016, de Sourceforge: http://vsqlbuilder.sourceforge.net/

Gallardo-Ruiz, J. E., & García-López, C. M. (2017). Apuntes para la asignatura Informática. Facultad de Ciencias (Matemáticas). *Diseño modular*. Málaga, Málaga, España: Dpto. Lenguajes y Ciencias de la Computación - UMA.

Gamma, E., Helm, R. J., & R. Vlissides, J. (2009). *Patrones de diseño.* Recuperado el 28 de febrero de 2017, de Patrones de diseño: http//www.vico.org/pages/PatronsDisseny.html..

García, J. (mayo de 2005). *Lenguaje de Modelado.* Recuperado el 3 de diciembre de 2016, de http://www.ingenierosoftware.com/analisisydiseno/uml.php

GoJS. (2017). *GoJS.* Recuperado el 15 de abril de 2017, de Interactive JavaScript Diagrams in HTML.: https://gojs.net.

Granados-Romero, J. (2015). *Las TIC, TAC, TEP, COMO INSTRUMENTO DE APOYO AL DOCENTE DE LA UNIVERSIDAD DEL SIGLO XXI.* Guayaquil - Mexico: Universidad de Guayaquil.

Gutiérrez, E. (octubre de 2009). JavaScript: Conceptos básicos y avanzados. Barcelona, España: Informática Técnica. Recuperado el 16 de abril de 2017

Hernandez Castillo, L. V. (abril de 2008). *Universidad Nacional Autónoma de México.* Recuperado el 4 de diciembre de 2016, de Facultad de Contaduría y Administración: http://fcasua.contad.unam.mx/apuntes/interiores/docs/98/4/informatica_4.pdf

Infante-Montero, S. (1 de abril de 2012). *Maestros del web.* (E. Tobar, Ed.) Recuperado el 27 de febrero de 2017, de Curso de Django para perfeccionistas: http://www.maestrosdelweb.com/guias/#guias-django

ITI. (2017). *Instituto Tecnológico de Informática*. Recuperado el 29 de abril de 2017, de Testeo de estrés y carga.: http://www.iti.es/servicios/servicio/resource/7240/index.html.

JetBrains Pycharm. (2017). *JetBrains*. Recuperado el 5 de diciembre de 2016, de Meet PyCharm: http://www.jetbrains.com/pycharm/quickstart/.

Kasampalis, S. (2015). *Mastering Python design patterns.* Birmingham, West Midlands, United Kingdom: Packt Publishing Ltd.

Larman, C. (2004). *UML y Patrones: una introducción al análisis.* Recuperado el 29 de enero de 2017, de UML y Patrones: http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf

REFENCIAS BIBLIOGRÁFICAS

Lozano, R. (12 de enero de 2011). *www.thinkepi.net.* Obtenido de http://www.thinkepi.net/las-tictac-de-las-tecnologias-de-la-informacion-y-comunicacion-a-las-tecnologias-delaprendizaje-y-del-conocimiento

Marqués-Andrés, M. M. (12 de febrero de 2001). *Herramientas CASE.* Recuperado el 4 de diciembre de 2016, de Universidad Jaume I: http://web.archive.org/web/20100928122456/http://www3.uji.es/%7Emmarques/f47/apun/node75.html

Martínez, J. C., Guzmán, L. A., Alarcón, V. M., & Gómez, C. M. (2013). Diagramas de navegación en aplicaciones Web. *Vol. 10(No. 2)*.

Méndez, A. V. (2010). *Metodologías de desarrollo de Software.*

Microsoft. (2017). *Revisiones de código y estándares de codificación.* Recuperado el 10 de Abril de 2017, de http://msdn.microsoft.com/es-es/library/aa291591%28v=vs.71%29.aspx

Ministerio de Educación Superior [MES]. (2014). *Plan de estudio "D" Ingeniería en Ciencias Informáticas.* La Habana.

Nielsen, J. (2010). *Usability Engineering.* San Francisco: Editorial Morgan Kaufman.

Ochoa P, L. A. (18 de enero de 2008). Proyecto del Máster Oficial de Software Libre. *Desarrollo de una interfaz gráfica para la creación de sentencias SQL: Caso base de datos PostgreSQL*. Barcelona, Catalunya, España: (Universitat Oberta de Catalunya. Recuperado el 20 de febrero de 2017

Ortíz, A. (23 de 4 de 2015). Lenguaje de base de datos. *Lenguaje de manipulación de datos*.

Pérez Porto, J., & Gardey, A. (2012). *Definicion.de*. Obtenido de Definición de SQL: http://definicion.de/sql/

pgAdmin. (29 de septiembre de 2016). *pgAdmin, PostgreSQL Tools.* Recuperado el 3 de diciembre de 2016, de https://www.pgadmin.org/

PostgreSQL Global Development Group. (2014). *PostgreSQL*. Recuperado el 5 de diciembre de 2016, de Sistema de gestión de base de datos relacional orientada a objetos y libre: http://www.postgresql.org/

PremiumSoft™ CyberTech Ltd. (2017). *Navicat*. Obtenido de Administrador de base de datos con interfaz gráfica de usuario para MySQL, MariaDB, SQL Server, SQLite, Oracle y PostgreSQL.: https://www.navicat.com/es

Pressman, R. (2010). *Ingeniería de Software, un enfoque práctico.* Nueva York.

Python. (2017). *Guía de estilo para el código Python – PEP 8 en Español.* Recuperado el 12 de Abril de 2017, de www.recursospython.com: www.recursospython.com/pep8es.pdf

REFENCIAS BIBLIOGRÁFICAS

Ramos, M. (2014). *McGraw-Hill Interamericana de España.* Recuperado el 5 de diciembre de 2016, de http://www.mcgraw-hill.es/bcv/guide/capitulo/8448148797.pdf

Rodríguez, S. (2012). *Módulo Informático para la Gestión de la Información Académica de los Estudiantes.* Matanzas.

Rossum, G. v. (febrero de 2017). *El tutorial de Python.* Obtenido de http://docs.python.org.ar/tutorial/pdfs/TutorialPython3.pdf

Rouse, M. (2014). *WhatIs.com - TechTarget*. Obtenido de Definition database (DB): http://searchsqlserver.techtarget.com/definition/database

Saldaña, G. (2017). *Nethazard*. Recuperado el 18 de abril de 2017, de http://blog.nethazard.net.

Sánchez, T. R. (6 de Marzo de 2015). Metodología de desarrollo para la Actividad productiva de la UCI v1.2. La Habana, Cuba.

Sommerville, I. (2011). *Ingeniería de Software.* México: Pearson Educación de México.

Torrealba, D. (8 de enero de 2016). *Principales librerías.* Obtenido de Filosofía de uso e importancia de las librerías en C++: https://sites.google.com/site/magisterdolorestorrealba/unidad-i-introduccion-a-el-lenguaje-c/principales-libreras

Vigo. (28 de noviembre de 2014). *Vigo: Unión Webmaster Libres*. Recuperado el 5 de diciembre de 2016, de IDE (Entornos de Desarrollo Integrado): https://sites.google.com/site/vigomiciudad/otras-cosa-interesantes/ide-entornos-de-desarrollo-integrado

Visual Paradigm. (15 de diciembre de 2016). *Visual Paradigm*. Obtenido de https://www.visual-paradigm.com/support/faq.jsp

Zaragosa, A. (3 de Abril de 2013). Recuperado el 2 de diciembre de 2016, de Scribd: https://es.scribd.com/document/49435158/Migrar-base-de-datos

ANEXOS

Anexo 1. Entrevista al Cliente

Estimado profesor: Se necesita de su cooperación en una investigación para una tesis de pregrado. Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Considera que los estudiantes logran desarrollar adecuadamente la habilidad diseñar implementar sentencias de manipulación de Bases de Datos Relacionales, como se refleja en su plan de estudio?
2. ¿Cuáles son las causas que podrían estar influyendo en el desarrollo de esta habilidad?
3. ¿Existe alguna herramienta en la Universidad para la implementación de sentencias manipulación de Bases de Datos Relacionales?
4. ¿Cuáles son sus características?
5. A partir de estas características, ¿considera que la herramienta posee lo necesario para implementar sentencias de acorde a lo establecido en la asignatura?
6. ¿Considera que el módulo de implementación de sentencias de manipulación de Bases de Datos Relacionales deba estar integrado a una plataforma educativa?
7. ¿La plataforma educativa *RDB-Learning* permite la implementación de sentencias?
8. ¿Cómo debería realizarse la revisión y evaluación de los ejercicios resueltos por los estudiantes en la plataforma *RDB-Learning*?
9. ¿Qué otras características, considera que deba presentar el sistema, en cuanto a la usabilidad, seguridad, interfaz u otro aspecto que garantice su calidad?

Muchas gracias por su colaboración.

Anexo 2. Encuesta para evaluar el nivel de aceptación de la solución

Estimado profesor: Se necesita de su cooperación en una investigación para una tesis de pregrado para evaluar el nivel de aceptación de una herramienta web, por ello, sería de gran ayuda que respondiera lo siguiente, con la sinceridad y seriedad que el proceso necesita.

Respecto al Módulo de implementación de sentencias de manipulación de bases de datos relacionales para la plataforma *RDB-Learning*, emita su valoración respecto a los siguientes parámetros: **MA**, si valora el parámetro como Muy Adecuado; **A**, si lo valora como Adecuado; **PA**, si considera que es Poco Adecuado; o **NA** si considera que es No Adecuado. Marcar con una X.

No	Parámetros	Valoración			
		MA	A	PA	NA
1	Tratamiento de los aspectos teóricos de la asignatura en el módulo de implementación de sentencias				
2	Nivel de apoyo que brinda al profesor el uso del módulo de implementación de sentencias para la orientación y evaluación de ejercicios				
3	Contribución del módulo al tema de implementar sentencias de manipulación de bases de datos relacionales en la asignatura SBD I				
4	Presentación de una interfaz agradable e intuitiva para el usuario				
5	Usabilidad del módulo				