Universidad de las Ciencias Informáticas

Facultad 4

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

Ambiente inteligente para el aprendizaje del lenguaje estructurado de consulta

(AIA-SQL)

Autores:

Yaritza Bárbara González Ramírez.

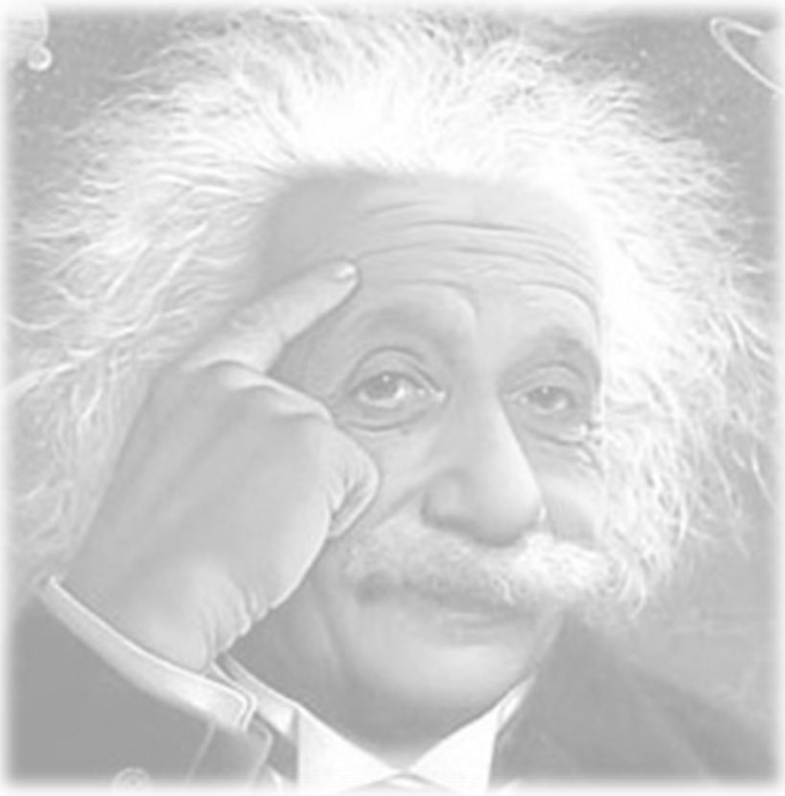
Osiel Menéndez García.

Tutores:

Dr. Enrique José Altuna Castillo.

Ing. Lisandra Guibert Estrada.

Ciudad de la Habana



"El genio se hace con un 1% de talento, y un 99% de trabajo."

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaramos que somos los autores del trabajo “Ambiente inteligente para el aprendizaje del lenguaje estructurado de consulta” y delegamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autores:

Yaritza B. González Ramírez.	Osiel Menéndez García.

Tutores:

Dr. Enrique José Altuna Castillo.	Ing. Lisandra Guibert Estrada.

DEDICATORIAS

A mis padres, a los que considero los mejores padres del mundo. Ellos han sido el impulso para llegar hasta aquí y así se sientan orgullos de su hija, quien siempre estará en deuda con ellos por todo lo que representan en su vida...

A la persona más importante en mi vida, mi hermano, por ser siempre un ejemplo para mí…

A toda mi familia por apoyarme en todo momento y confiar siempre en mí.

Yaritza B. González Ramírez.

A mis padres por mostrarme el camino correcto.

A mi famila en general, por brindarme siempre su apoyo.

Osiel Menéndez García.

AGRADECIMIENTOS

Todo mérito conlleva sacrificios pero el resultado te complace siempre que lo logres a pesar de los obstáculos, a pesar de caer sin saber en ocasiones como levantarte. Pero para nada te satisface el crédito si no lo disfrutas en compañía de los que aprecias y si de alguna manera les haces saber que todo se lo debes a ellos por eso quiero agradecerles a:

Mi familia, mis padres por mostrarme el camino correcto, por ser ejemplo y por la educación que me han dado.

Mis amistades aunque con sus virtudes y defectos fueron mi familia estos 5 años.

Mi compañera que entre molestias y discusiones logramos juntos que esta tesis se realizara con éxito y por ser de gran apoyo y buena amiga.

Mis tutores Lisandra y Enrique que han sido guías claves en el desarrollo de este trabajo y sin ellos no hubiera sido posible el resultado.

Osiel Menéndez García.

AGRADECIMIENTOS

A mis padres, por su apoyo incondiconal, por guiarme en la vida con paciencia y dedicación, por enseñarme a luchar por mis sueños. Gracias por confiar en mi y porque sin ustedes este momento no hubiese sido posible.

A mi cuñada y a mi hermano que a pesar de la distancia que nos separa, hoy lo siento más cerca que nunca. A ambos por su apoyo, su cariño, sus consejos y por haberme regalado lo más lindo que tengo en la vida, la dicha de ser tía.

A toda mi familia por su apoyo incondicional y estar siempre pendientes de mis estudios y resultados, muy especial a mis tios Belkis y Fidel a los cuales considero como mis padres.

A mi novio, gracias por estar siempre a mi lado, gracias por tu comprensión, por tu amor y tu cariño y gracias por levantarme el ánimo y secar mis lágrimas cuando creía que no iba a lograrlo.

A mi familia de Pinar, que desde el primer día que llegue a sus vidas me acogieron como una más de sus miembros, regalándome momentos inolvidables, gracias a todos por su apoyo y su cariño, de más está decir cuan importante son en mi vida.

A mi compañero de tesis por soportar mi carácter, por todo su apoyo y comprensión y por ayudar a que este sueño se hiciese realidad.

A mis tutores Lizandra y Enrique por ayudarnos y guiarnos durante toda la investigación dedicándonos todo su tiempo sin importar cuán ocupados estuviecen.

A mis hermanitas del alma, dos personitas super especiales a las cuales adoro con la vida, gracias por sus regaños cuando los merecía, por su cariño, por su amistad y por regalarme un pedacito de su corazón. Esto es para ustedes Yanet y Nayara.

A mis amigos y amigas con los cuales he compartido momentos inolvidables y difíciles, a todos les agradezco su apoyo, su cariño y su respeto, sepan que los quiero mucho a todos.

En fin gracias a todos los que estuvieron presentes a mi lado a lo largo de este sueño, hoy hecho realidad.

Yaritza B. González Ramírez

RESUMEN

El desarrollo alcanzado en la actualidad por las Tecnologías de la Información y las Comunicaciones (TIC) ha potenciado un gran impacto en todas las esferas de la vida, principalmente, en el Proceso de Enseñanza-Aprendizaje (PEA). El mismo, constituye un proceso que se encuentra en constante cambio en gran medida debido al avance adquirido por las TIC, siendo fundamental para su desarrollo la intervención de los profesores y estudiantes. Actualmente durante las actividades docentes, los alumnos asumen un rol más activo haciendo uso de contenidos y herramientas digitales creadas para apoyar el PEA. El objetivo de la presente investigación, es desarrollar un ambiente inteligente que brinde soporte a los estudiantes durante el aprendizaje del lenguaje estructurado de consulta (SQL) a través de la evaluación automática de consultas y la entrega de retroalimentación. Para su cumplimiento, se realiza un estudio sobre los sistemas destinados al aprendizaje de SQL, los Jueces en Línea, los Sistemas Tutores Inteligentes y la retroalimentación que estos últimos generan. Además se seleccionaron justificadamente las tecnologías, herramientas, lenguajes y metodología a utilizar en el desarrollo del sistema, obteniéndose como resultado una solución que cumple con los objetivos planteados.

Palabras Claves: SQL, Juez en Línea, Sistema Tutor Inteligente, retroalimentación.

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad, las Tecnologías de la Información y las Comunicaciones (TIC) han alcanzado un vertiginoso desarrollo, convirtiéndose en el eslabón fundamental para alcanzar el éxito en diferentes esferas de la vida. La educación no ha quedado exenta de este progreso, evidenciándose en la estrecha relación existente entre el Proceso de Enseñanza–Aprendizaje (PEA) y el avance adquirido por las TIC, siendo estas últimas en la mayoría de los casos, un medio de apoyo tanto para el profesor como para sus estudiantes.

La incorporación de las TIC en las escuelas ha venido relacionada tradicionalmente con la tecnología, aunque varios son los factores que entran en juego para un buen aprendizaje asistido por computadora. En primer lugar se necesita disponer de la tecnología apropiada, pero no basta con tener un buen hardware en el aula para trabajar satisfactoriamente, sino que cada vez se hace más necesario disponer de contenidos digitales (software) de cada materia que el profesor pueda utilizar y manejar de acuerdo con sus necesidades. Y por supuesto, para dar cohesión a todo lo anterior, la figura del profesor se convierte en el factor determinante como dinamizador, orientador y asesor de todo el PEA. (Polanco, 2007)

El uso de las TIC en apoyo al PEA ha proporcionado nuevos canales de comunicación que se deben aprovechar. Estas fomentan la colaboración entre los alumnos, ayudan a centrarse en los aprendizajes, aumentan la motivación y el interés, favorecen la búsqueda, estimulan el desarrollo del razonamiento, la resolución de problemas, la creatividad y la capacidad de aprender a aprender. (Polanco, 2007)

La Universidad de las Ciencias Informáticas (UCI) desde sus inicios, se ha encargado de formar profesionales calificados para producir servicios informáticos que generalmente surgen como apoyo al PEA. Actualmente en la UCI como parte del programa de estudio, se imparte la asignatura Sistemas de Bases de Datos y es en ella donde los estudiantes matriculados en la carrera de Ingeniería en Ciencias Informáticas, enfrentan por primera vez el paradigma de Programación Funcional. Específicamente, basándose en el aprendizaje del Lenguaje Estructurado de Consulta (en inglés SQL), en el cual se necesita diseñar sentencias que obtengan los datos, que luego se procesan para resolver un problema.

Luego de consultar diferentes bibliografías, se ha podido llegar a la conclusión de que las apreciaciones de diferentes autores que han definido el concepto de **SQL**[1] son muy diversas. Ejemplo de ello es la que a continuación se cita: "*lenguaje estandarizado, diseñado para acceder y manipular los datos almacenados en bases de datos relacionales y para trabajar con las propias bases de datos*" (Darie, 2003).

[1] **SQL:** Structured Query Language (Lenguaje estructurado de consulta.)

A pesar de su importancia, durante la enseñanza de este conocimiento persisten algunas deficiencias, ya que no existe en la universidad ninguna aplicación que brinde soporte durante la realización de las actividades que les son propuestas a los estudiantes. Por otra parte, son insuficientes los recursos didácticos utilizados para orientar a los alumnos cómo abarcar todo lo que necesitan conocer acerca de este importante tema en la asignatura, estudiar de manera independiente y auto-evaluarse.

Si bien son variados los intentos de solución al problema de automatizar la evaluación cabe destacar a los Jueces en Línea, aunque presentan un limitado nivel de profundidad en la retroalimentación siendo esta su principal desventaja. Presentándose, los Sistemas Tutores Inteligentes como una solución a esta dificultad.

Al hablar de un **Juez en Línea** según (Vázquez, 2012) se hace referencia a: "(…) *toda aplicación web para entornos generalmente académicos, que incluye varios problemas de distintas materias para ser resueltos mediante técnicas de programación y, lo más importante, evalúa automáticamente las soluciones de sus usuarios en los varios lenguajes de programación disponibles*". En el caso particular de los **Tutores Inteligentes** son definidos en varias bibliografías como "*sistema de software que utiliza técnicas de Inteligencia Artificial para representar el conocimiento e interactúa con los estudiantes para enseñárselo*" (Hillsdale, 2009).

El uso de la tecnología y la aplicación de sistemas inteligentes en ambientes de aprendizaje nos conducen a un modelo de enseñanza-aprendizaje en el cual los alumnos asumen un rol más activo. De forma general, se observa que los escolares que trabajan en entornos de aprendizaje potenciados con agentes tutores demuestran gran mejoría en las diferentes áreas del conocimiento. (Sleeman, 1982)

En el entorno mundial se encuentran un elevado número de herramientas erigidas para brindar ayuda en el aprendizaje de este tema, pero sin dudas, entre las más destacadas y utilizadas se encuentran LEARN-SQL (Learning Environment for Automatic Rating Notions of SQL), Computer Assisted Assessment of SQL query Skills, Learning SQL with a computerized Tutor, SQLator: an online SQL learning workbench y un Sistema Tutor para la enseñanza del lenguaje SQL (SQL-Tutor). Estas aplicaciones comparten la característica de ser sistemas que sólo permiten la corrección automática de ejercicios. No obstante, se limitan a corregir el tipo más básico de ejercicios posibles.

Cuba es un país que no ha quedado exento del avance tecnológico, llevando a cabo el desarrollo de varios sistemas para el apoyo del aprendizaje de los cuales muchos han sido desarrollados en la UCI, precisamente sobre la temática en cuestión se encuentra el Juez en Línea de Bases de Datos (Data Base Online Judge) pensado con el objetivo de evaluar el código programado en lenguaje de Base Datos para apoyar a los estudiantes y profesores de las asignaturas de Base de Datos I y II. (Rodés, 2010)

La utilización de este sistema brinda grandes ventajas para el aprendizaje de este tema debido a que permite la evaluación automática del código programado en SQL, pero a su vez no ofrece la posibilidad de brindar información específica sobre los errores al estudiante.

Por todo lo antes planteado se establece como **problema a resolver** ¿Cómo brindar soporte a los estudiantes durante el aprendizaje del lenguaje estructurado de consultas en la asignatura de Sistemas de Bases de Datos?. De lo planteado anteriormente se deriva como **objeto de estudio** el proceso de enseñanza del lenguaje estructurado de consulta. Determinando para la investigación, como **campo de acción** el proceso de enseñanza del lenguaje estructurado de consulta para el apoyo del tema Programación Funcional en la asignatura Sistema de Bases de Datos.

El **objetivo general** que se persigue con la investigación es desarrollar un ambiente inteligente para la enseñanza del lenguaje estructurado de consultas, que brinde soporte a los estudiantes durante el aprendizaje en la asignatura de Sistemas de Bases de Datos.

Para darle respuesta al objetivo general propuesto se definen los siguientes **objetivos específicos**:

- Elaborar el marco teórico de la investigación donde se relacionen los principales referentes relativos al proceso de enseñanza del lenguaje estructurado de consulta.
- Realizar el flujo de análisis y diseño con el objetivo de identificar los elementos arquitectónicos y los elementos de diseño para la posterior implementación.
- Implementar las funcionalidades correspondientes a la herramienta.
- Probar las funcionalidades correspondientes a la herramienta mediante la aplicación de las pruebas de caja negra.

Definiéndose para la investigación las siguientes **tareas de investigación**:

- Estudiar la bibliografía actualizada para la elaboración del marco teórico-conceptual referente a los sistemas para el apoyo a la enseñanza de lenguaje estructurado de consultas.
- Identificar las limitaciones de los sistemas similares a la solución que se propone.
- Identificar los requerimientos con los que debe cumplir la herramienta.
- Generar los artefactos correspondientes con la metodología de desarrollo de software seleccionada.
- Definir los elementos que componen la arquitectura de la herramienta a implementar.
- Implementar las funcionalidades para la herramienta.
- Realizar las pruebas a la herramienta para verificar la correspondencia de las funcionalidades con

los requerimientos identificados.

Defendiendo como **hipótesis científica**: Si se desarrolla un ambiente inteligente para el aprendizaje del lenguaje estructurado de consultas, se obtendrá un sistema informático que brinde soporte a los estudiantes durante el desarrollo de las actividades y de esta forma, servirá de apoyo al proceso de enseñanza-aprendizaje de dicho tema en la asignatura Sistema de Bases de Datos.

Los **resultados** esperados con esta investigación son: realizar una correcta documentación del producto, garantizando con esta el futuro mantenimiento del sistema y desarrollar un sistema informático que permita la evaluación automática de consultas a bases de datos y a su vez ofrezca soporte al estudiante durante la realización de las actividades.

Los métodos científicos utilizados en la investigación estuvieron determinados por el objetivo general y los objetivos específicos. A nivel teórico fueron utilizados los métodos: **analítico-sintético**, para realizar un estudio bibliográfico profundo de la teoría existente alrededor del objeto de estudio, y a partir del mismo determinar las características que tendrá la solución propuesta; **inductivo-deductivo**, para el estudio de las principales iniciativas de aplicaciones destinadas al aprendizaje del lenguaje estructurado de consulta, con el objetivo de determinar cuáles son las alternativas viables a incorporar en la presente investigación; **análisis documental**, en la consulta de la bibliografía especializada en los tópicos afines a la investigación y el **hipotético-deductivo**, en la elaboración de la hipótesis de la investigación y en la propuesta de nuevas líneas de trabajo a partir de los resultados obtenidos. También fueron utilizados métodos empíricos, como es el caso de la **observación**, con el objetivo de estudiar de cerca cómo funciona el proceso de aprendizaje del lenguaje estructurado de consulta en la universidad y los principales problemas asociados a este.

El presente trabajo posee la siguiente estructura:

Capítulo I: Se describe el marco teórico de la investigación realizada. Se investigan y analizan los componentes relacionados con el estado del arte del proceso de enseñanza del lenguaje estructurado de consulta y su evolución nacional e internacional. Además, se analizan las distintas herramientas, tecnologías y metodología a utilizar.

Capítulo II: Se especifica la propuesta de solución para el problema planteado. Se presenta el Modelo de Dominio y se exponen los principales requisitos funcionales y no funcionales que debe cumplir el sistema. Además se obtienen como artefactos fundamentales los casos de uso arquitectónicamente significativos y sus descripciones.

Capítulo III: Se realiza el análisis y diseño de la propuesta de solución planteada. Se generan los

diagramas de clases del análisis y el diseño, el modelo de Bases de Datos y el modelo de despliegue. Además se realiza un estudio de los patrones de arquitectura y diseño empleados.

Capítulo IV: Se obtiene la descripción del proceso de implementación de la herramienta a través de los diagramas de componentes. Por último se realiza la descripción de los casos de prueba y se describen los resultados obtenidos una vez aplicadas las pruebas.

CAPÍTULO I

Capítulo 1: Fundamentación teórica

Introducción

En este capítulo se abordan los principales conceptos referentes a la investigación así como los principales aspectos investigados para sustentar el desarrollo de dicho sistema. Además se realiza el estudio de algunas herramientas similares existentes relacionadas al objeto de estudio. Finalmente, se seleccionan y caracterizan las herramientas, metodología de desarrollo de software y los lenguajes de modelado y de programación a utilizar en la aplicación.

1.1 Fundamentos Teóricos y Metodológicos

El lenguaje estructurado de consulta (**SQL**) tiene una amplia aceptación y es utilizado en el trabajo diario con una base de datos relacional. Dada la importancia del lenguaje SQL se han desarrollado distintas herramientas que ofrecen soporte a su aprendizaje. Con este fin, son muy utilizados los **Jueces en Línea** y los **Tutores Inteligentes**. (Masó, 2010)

1.1.1 Lenguaje Estructurado de Consulta

El lenguaje estructurado de consulta (SQL), es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre la misma.

Las apreciaciones de diferentes autores que han definido el concepto de SQL son diversas aunque no difieren entre sí, "*un lenguaje estandarizado diseñado para acceder y manipular los datos almacenados en bases de datos relacionales y para trabajar con las propias bases de datos*" (Darie, 2003). En otras fuentes se puede encontrar como "*tipo de lenguaje vinculado con la gestión de bases de datos de carácter relacional que permite la especificación de distintas clases de operaciones entre éstas. Gracias a la utilización del álgebra y el cálculo relacional, SQL brinda la posibilidad de realizar consultas con el objetivo de recuperar información de las bases de datos de manera sencilla*" (Coca, 2011). Otras bibliografías lo definen como *"lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en estas.*" (Miranda, 2004)

A continuación se enuncian algunas de las **características generales** de dicho lenguaje propuestas en (Darie, 2003):

- Es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.
- Es un lenguaje declarativo de alto nivel o de no procedimiento, que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación. De esta forma una sola sentencia puede equivaler a uno o más sentencias de código en un lenguaje de bajo nivel.
- Manejo del álgebra y el cálculo relacional permitiendo efectuar consultas con el fin de recuperar de una forma sencilla información de interés de una base de datos, así como también hacer cambios sobre ella.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado así como un conjunto de operadores que se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

En la actualidad, es posible desarrollar entornos en línea que faciliten la evaluación de distintos temas. Un entorno común y muy utilizado en este contexto son los jueces en línea cuya función principal es evaluar el código que fue enviado a ellos. Esta evaluación genera respuestas como: correcto, incorrecto, error de compilación, error de ejecución, entre otras. (Ribeiro, 2011)

1.1.2 Jueces en Línea

Los **Jueces en Línea** son definidos en (Vázquez, 2012) como "*toda aplicación web para entornos generalmente académicos, que incluye varios problemas de distintas materias para ser resueltos mediante técnicas de programación y, lo más importante, evalúa automáticamente las soluciones de sus usuarios en los varios lenguajes de programación disponibles*". Otros autores los definen como "*sistemas que compilan y ejecutan un código fuente y realizan la evaluación automática de estos códigos*". (Ferreira, 2013)

Un juez en línea presenta varias ventajas con respecto a otras aplicaciones de escritorio concebidas para la automatización de la evaluación, debido al hecho de ser aplicaciones en línea (Vázquez, 2012):

Portabilidad: Al presentar una interfaz construida a base de estándares como HTML[2], pueden ser ejecutadas desde cualquier plataforma, con el único requisito de contar con un navegador que soporte dichos estándares.

Accesibilidad: Al estar publicadas en un servidor de Internet en el cual guardan todos sus datos y configuración, generalmente pueden ser accedidas desde cualquier computadora con acceso a la red de redes.

Colaboración: Al centrar todas las conexiones de los usuarios en un mismo servidor central, se facilita la colaboración y comunicación entre los mismos, haciendo énfasis en el trabajo en equipo y la distribución de la información.

En (Kurnia, 2001) se hace una de las mejores descripciones en lo que refiere a un modelo de juez en línea. En el mismo se describen las múltiples desventajas de la evaluación manual cargadas de subjetivismo y dependientes de las circunstancias en que se lleva a cabo la evaluación, así como las también múltiples ventajas de la evaluación automática que reduce notablemente el tiempo empleado para realizar esta tarea. (Vázquez, 2012)

A continuación se presenta una descripción propuesta en (Vázquez, 2012) de la manera en que un juez en línea comprueba la corrección de un programa:

Para cada problema a resolver existen uno o varios ficheros que representan las entradas para probar la corrección de las soluciones enviadas por los usuarios, a su vez existen igual cantidad de ficheros que constituyen las salidas correctas asociadas a cada una de esas entradas. Se compila la solución enviada por el usuario, escogiendo de manera transparente el compilador a utilizar según el lenguaje de programación utilizado. De no haber errores de compilación, el paso siguiente es ejecutar el programa para cada entrada definida y verificar que la salida de cada ejecución coincida exactamente con la salida correcta definida previamente para la entrada en cuestión. La solución enviada se considerará correcta si cada una de estas verificaciones fue exitosa y en caso contrario se considerará incorrecta.

Para garantizar un veredicto objetivo es muy importante que las entradas de prueba sean lo suficientemente abarcadoras teniendo en cuenta todos las posibles situaciones que se puedan presentar según el problema, es importante recalcar que *correcta* e *incorrecta* no son los únicos veredictos que brinda un juez en línea. En la figura 1 se representa el esquema general de funcionamiento de un juez en línea.

[2]**HTML:** Hypertext Markup Language.

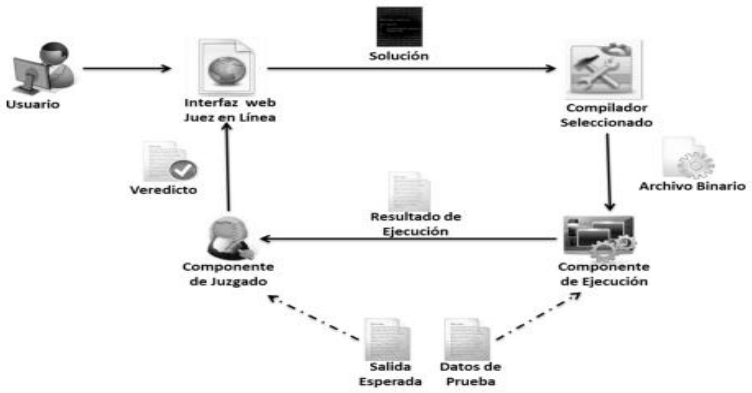


Figura 1 Esquema de funcionamiento general de los jueces en línea. (Vázquez, 2012)

Si bien las respuestas instantáneas del juez en línea promueven el aprendizaje efectivo, algunos estudiantes se frustran si no son capaces de obtener todas las respuestas correctas. La corrección de un programa que se basa únicamente en la ejecución de varios conjuntos de datos también puede intimidar a algunos estudiantes. Estas frustraciones y miedos pueden ser parcialmente aliviados al proporcionar una biblioteca con ejemplos de problemas viejos destinados a los estudiantes para acostumbrarse al sistema. (Ribeiro, 2011)

En los jueces en línea existentes, se observa que el desarrollo está dirigido para su uso en competencias y por lo tanto algunas de sus funciones didácticas se encuentran en ellos. Por ejemplo, después de la presentación de la resolución de un problema, el usuario obtiene un resultado que sólo indica si la respuesta es correcta/incorrecta o si hubo errores en tiempo de ejecución/construcción. Este resultado, sin embargo, no proporciona más información que pudiera indicar al usuario que salió mal como encontrar su error y solucionarlo. (Ribeiro, 2011)

La principal dificultad de los Jueces en Línea es que no cuentan con un mecanismo de orientación a los estudiantes. Los Tutores Inteligentes constituyen una posible solución a esta dificultad.

1.1.3 Sistemas Tutores Inteligentes

Los **Sistemas Tutores Inteligentes (STI)** comenzaron a desarrollarse en los años ochenta con la idea de poder impartir el conocimiento usando alguna forma de inteligencia para poder asistir y guiar al estudiante en su proceso de aprendizaje (Orlowska, 2004). Los mismos tienen como objetivo la enseñanza personalizada, incorporando técnicas de Inteligencia Artificial (IA) en la generación de rutas de aprendizaje, selección de actividades, soporte durante las actividades, evaluación, presentación de los contenidos, entre otras. (Woolf, 2009).

Más formalmente (Wenger, 1987) define: "*Un STI utiliza técnicas de IA, principalmente para representar el conocimiento, y dirigir una estrategia de enseñanza; y es capaz de comportarse como un experto, tanto en el dominio de conocimiento que enseña como en el dominio pedagógico, donde es capaz de diagnosticar la situación en la que se encuentra el estudiante, y de acuerdo a ello, ofrecer una acción o solución que le permita progresar en el aprendizaje*". Giraffa los delimita como: "*un sistema que incorpora técnicas de IA a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa"* (Giraffa, 1997). Por su parte Wolf define los STI como: "*sistemas que modelan la enseñanza, el aprendizaje, la comunicación y el dominio del conocimiento del especialista y el entendimiento del estudiante sobre ese dominio*". (Woolf, 1984)

El propósito del STI es presentar un comportamiento similar al de un tutor humano, que se adapte a las necesidades del estudiante, identificando la forma en que el mismo resuelve un problema para poder brindarle ayuda cuando cometa errores (Zulma, 2010). Una de las metas de un STI es contribuir con la apropiación de conocimientos, habilidades y actitudes, para ello deberá incorporar ciertas "cualidades" que emulen de forma aproximada las buenas prácticas docentes, alguna de ellas según (Perrenoud, 2004) son: guiar, orientar, motivar, problematizar, provocar y apoyar el recorrido del alumno, diseñar situaciones y ofrecer las ayudas adecuadas para la superación de los obstáculos en el aprendizaje.

Las características principales de un STI son: promover una respuesta activa en el alumno, informar al alumno sobre su desempeño, permitir un avance del aprendizaje de manera autónoma, promover la eficiencia y eficacia del alumno en el trabajo. Además, deben ser inteligentes en comparación con los sistemas tradicionales de instrucción por computadora, así como poseer la capacidad tanto para resolver un problema que se le presenta a un alumno como también la capacidad de explicar cómo lo resolvió. Es claro que los STI son catalogados como herramientas muy efectivas para facilitar a los alumnos el PEA, debido a que por medio de ellos se facilita a los estudiantes los contenidos. (Arias, 2007)

La arquitectura general de un STI está conformada por cuatro componentes que constituyen los tipos de conocimientos que participan en la tutoría: modelo de dominio, modelo del alumno, modelo pedagógico y modelo de interfaz. (Polson, 1988)

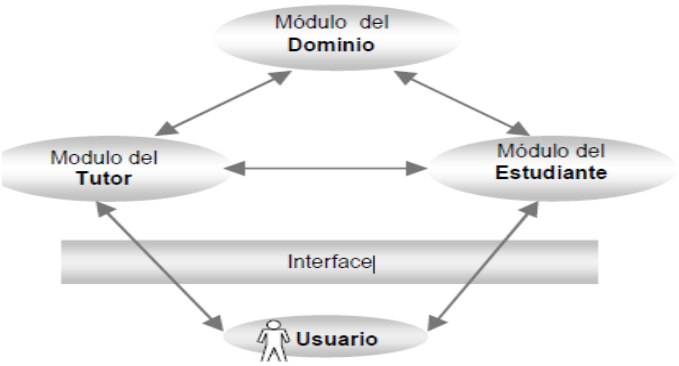


Figura 2 Interacción de los Módulos de un Sistema Tutor Inteligente. (Polson, 1988)

Cada uno de estos componentes asume distintas funciones interactuando entre sí. A continuación basados en (Guzman, 2005) presentaremos una descripción de cada uno de ellos:

El ***modelo de dominio*** corresponde a la respuesta sobre el *qué se enseña.* El ***modelo del alumno o estudiante*** representa a *quién se enseña*. La mayoría de los STI infieren este modelo a partir de los conocimientos y carencias del alumno sobre el modelo del dominio, y a partir de esta información, adaptan el proceso de instrucción a sus necesidades. El ***modelo pedagógico o modelo de instrucción*** corresponde a la función *cómo se enseña*. Constituye por tanto, las estrategias de enseñanza o estrategias tutoriales es decir, cómo el sistema debe presentar el material educativo al alumno (Capps, 1988). Por otra parte la *interacción hombre/máquina* se realiza en el ***modelo de interfaz.***

1.1.4 Modelo del dominio

El modelo del dominio contiene las formas de representación y razonamiento que le permiten funcionar como una fuente de conocimiento y un estándar para evaluar las acciones del estudiante dentro de la temática que se aprende. La forma para enfocar el desarrollo de un modelo de dominio está directamente relacionada con la estructura de sus conocimientos y la idoneidad para soportar los procesos de tutoría (Castillo, 2014). Además contiene el conocimiento sobre la materia que debe ser aprendida, consiste en la creación de un sistema capaz de inferir conocimiento para verificar evaluaciones y recomendar contenido, según la planificación de los temas a desarrollar. Básicamente representa las explicaciones y el conocimiento que domina el docente en calidad de experto en su área.

El primer paso en la implementación de un STI, es la representación explícita por parte del experto del conocimiento existente sobre el dominio. Asimismo, es el encargado de la generación de problemas, y la evaluación de la corrección de las soluciones suministradas por el alumno. Los modelos del dominio pueden clasificarse a su vez en (Castillo, 2014):

- ✓ *Modelo de caja negra*: Es un medio de razonar sobre el dominio que no requiere una codificación explícita del conocimiento subyacente.
- ✓ *Modelo basado en la metodología de los sistemas expertos*: Se sigue los mismos pasos que en el desarrollo de un sistema experto. Implica extraer el conocimiento de un experto y decidir el modo en el que va a ser codificado y aplicado.
- ✓ *Modelo cognitivo:* El modelo del dominio se obtiene abstrayendo el modo en el que los humanos hacen uso del conocimiento. Este tipo de modelos es el más efectivo desde el punto de vista pedagógico, aunque requiere un mayor esfuerzo de implementación. Su estructura estaría condicionada al tipo de conocimiento que se quiera representar.

Luego de consultar diferentes bibliografías se identificaron tres formas fundamentales que se usan para representar y razonar sobre el conocimiento del dominio: modelos basados en reglas, basados en restricciones y sistemas expertos. (Castillo, 2014)

Los **modelos basados en reglas** son adecuados para ser usados en dominios donde las soluciones de los ejercicios puedan ser estructurados en etapas bien definidas y reconocibles, caminos de solución identificables y verificables (Vanlehn, 2005). Están compuestos por reglas que son aplicadas por los estudiantes, tanto correctas como incorrectas, de esta forma el sistema es capaz de ofrecer la retroalimentación que está codificada en la regla según la acción realizada. (Castillo, 2014)

Los **modelos basados en restricciones** expresan las condiciones que deben cumplir todas las soluciones correctas usando restricciones. El sistema diagnostica los errores por la violación de restricciones y entrega retroalimentación según la restricción que fue violada. (Mitrovic, 2012)

Los modelos antes mencionados son conocidos comúnmente como modelos cognitivos (Castillo, 2014). Estos pueden contener una descripción detallada y precisa del conocimiento que es usado por el estudiante, incluyendo estrategias y principios, así como forma para aplicarlos en el contexto de problemas específicos. (Mitrovic, 2009)

Los **modelos expertos** resuelven problemas mediante conocimiento codificado que ha sido extraído del análisis de las acciones o con otras técnicas, a partir de expertos reales del dominio (Moritz, 2008). La principal limitación de los sistemas expertos es la dificultad para justificar las inferencias realizadas o proporcionar explicaciones que sean apropiadas para el proceso de aprendizaje. Algunas de las técnicas que se han usado para la construcción de este tipo de modelo de dominio son: reglas de producción, redes neuronales, árboles de decisión y razonamiento basado en casos. (Castillo, 2014)

Como resultado del uso de STI se reportan logros significativos en el aprendizaje con respecto a otros tipos de sistemas que se usan para apoyar el PEA (Brawner, y otros, 2011). La causa principal de estos resultados positivos es la retroalimentación que recibe el estudiante durante la realización de actividades (Aleven, 2010), la misma se define para la investigación tomando como referencia a (Castillo, 2014) como: "*información que es mostrada a los estudiantes durante las actividades y que puede proporcionar a su vez información sobre los errores y estrategias para mejorar la solución*".

1.1.5 Retroalimentación en los STI basados en restricciones

La retroalimentación constituye una de las partes esenciales de los STI, debido a que durante el desarrollo de las actividades donde los estudiantes deben solucionar problemas aplicando los diferentes conocimientos es de vital importancias el uso de la misma, representándose esta última como información en forma de ayuda que se brinda al estudiante sobre los errores cometidos. (Bratko, 2014)

En la retroalimentación que se genera al estudiante se incluyen aspectos semánticos, relativos a elementos específicos de la actividad que se intenta solucionar, principalmente a través de las restricciones. (Mitrovic, 1998)

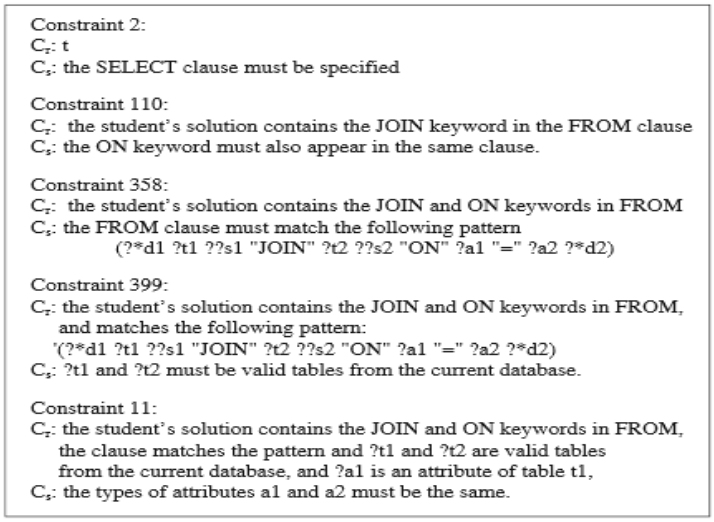
Cada restricción consiste en un par ordenado (Cr, Cs), donde Cr es la condición relevancia y Cs es la condición de satisfacción. La condición de relevancia comprueba si la restricción es aplicable a la solución del estudiante. Por ejemplo, la restricción podría ser aplicable a situaciones en las que el estudiante ha añadido dos fracciones con el denominador común. La condición de satisfacción especifica una prueba adicional que se debe cumplir por las soluciones correctas; para el mismo ejemplo, la solución del estudiante es correcta si el denominador de la fracción resultante es igual a los denominadores de las dos fracciones dadas, y el numerador es igual a la suma de los numeradores de las fracciones dadas. Si se cumple la condición de relevancia, pero no se cumple la condición de satisfacción, entonces la solución del estudiante es incorrecta (Mitrovic, 2012). Por lo tanto, la forma general de una restricción es:

Si<condición de relevancia>*es verdadera*,
Entonces<condición de satisfacción>*debe ser verdadera también*

El origen de un error no es crucial para la generación de intervenciones pedagógicas; el sistema de tutoría no tiene que ser capaz de reproducir el error, o entender la forma en que se generó con el fin de reaccionar de forma inteligente. Si se viola una restricción, estos pueden proporcionar información al estudiante sobre el principio de dominio que fue violada, sin saber exactamente qué tipo de conocimiento incorrecto ha causado el error. (Mitrovic, 2012)

En este punto, es necesario señalar la diferencia entre las restricciones y los reglas de producción. Aunque la declaración anterior parece similar a una regla de producción SI-ENTONCES, es de una naturaleza muy diferente. La parte SI de una regla de producción especifica las características de una situación a la que la regla es aplicable, y el objetivo a alcanzar, mientras que la parte ENTONCES especifica la acción a realizar si se cumplen tanto el objetivo y la situación. Por el contrario, una restricción especifica las condiciones para que un estado solución sea correcta. En el caso específico de las restricciones, las dos condiciones no están vinculadas con implicación lógica, pero si con el "debe" conectivo, como en si es cierto Cr, Cs debe ser cierto también (Ohlsson, 2007). Las reglas de producción son de carácter generativo, mientras que las restricciones son evaluativo, y se pueden utilizar para hacer juicio. (Mitrovic, 2012)

El conjunto de restricciones para un dominio dado representa explícitamente las características de todas las soluciones correctas. Cualquier solución que viola una o más restricciones es incorrecta. A continuación se muestra un ejemplo de restricciones definidas en (Mitrovic, 2003):


Constraint 2:
C_r: t
C_s: the SELECT clause must be specified

Constraint 110:
C_r: the student's solution contains the JOIN keyword in the FROM clause
C_s: the ON keyword must also appear in the same clause.

Constraint 358:
C_r: the student's solution contains the JOIN and ON keywords in FROM
C_s: the FROM clause must match the following pattern
(?*d1 ?t1 ??s1 "JOIN" ?t2 ??s2 "ON" ?a1 "=" ?a2 ?*d2)

Constraint 399:
C_r: the student's solution contains the JOIN and ON keywords in FROM, and matches the following pattern:
'(?*d1 ?t1 ??s1 "JOIN" ?t2 ??s2 "ON" ?a1 "=" ?a2 ?*d2)
C_s: ?t1 and ?t2 must be valid tables from the current database.

Constraint 11:
C_r: the student's solution contains the JOIN and ON keywords in FROM, the clause matches the pattern and ?t1 and ?t2 are valid tables from the current database, and ?a1 is an attribute of table t1,
C_s: the types of attributes a1 and a2 must be the same.

Figura 3 Ejemplo de restricciones. (Mitrovic, 2003)

Luego del análisis de diferentes bibliografías se concretó la existencia de alrededor de 400 restricciones. Con el objetivo de recopilar las mismas, se mantuvo contacto con Antonija Mitrovic quien notifica que las restricciones son privativas.

En la actualidad son diversas las herramientas desarrolladas con el objetivo de brindar apoyo al PEA, específicamente en el aprendizaje del lenguaje estructurado de consulta. En muchos casos su objetivo es simplemente que el alumno practique y se ejercite en la resolución de consultas. (Masó, 2010)

1.2 Análisis de las soluciones existentes

Las soluciones varían desde las más específicas a las más generales, es por ello que se realizó el análisis de soluciones similares existentes, el cual se muestra a continuación:

1.2.1 Soluciones similares existentes a nivel internacional

A nivel mundial se encuentran numerosos sistemas informáticos para apoyo del PEA existiendo diferencias entre estos, que están dadas en gran medida por las funcionalidades que brindan.

SQLator es una herramienta web interactiva para el aprendizaje de SQL. La principal aportación de esta herramienta es que dispone de una función para la corrección automática de SELECTS basada en complejos algoritmos heurísticos. No aporta retroalimentación alguna, solamente se limita a indicar si son correctas o incorrectas las soluciones enviadas por el alumno (Masó, 2010). El componente más importante del sistema es la equivalencia del motor SQLator, capaz de juzgar si una solución propuesta en SQL correspondiente a la declaración del estudiante es la correcta. (Sadiq S, 2004)

SQL-Tutor es un sistema de enseñanza basado en el conocimiento que enseña SQL a los estudiantes, el mismo analiza la solución del estudiante para que coincidan con las limitaciones y la solución ideal. Una característica muy importante es su simplicidad computacional (Mitrovic, 2003). Está diseñado como un entorno de prácticas en donde se supone que los alumnos ya conocen la materia y se ofrece como un complemento a las clases presenciales. Se ha adaptado la versión inicial para que sea una herramienta web. SQL-Tutor únicamente permite trabajar con sentencias SELECT. El sistema dispone de un conjunto de problemas para distintas bases de datos y la solución ideal para cada caso. La corrección se realiza a través de la comparación con una solución ideal. (Masó, 2010)

LEARN-SQL es una herramienta para el soporte al aprendizaje semipresencial en el ámbito de las bases de datos. El objetivo de dicha herramienta es corregir automáticamente cualquier tipo de sentencia SQL y discernir si la respuesta aportada por el estudiante es o no correcta con independencia de la solución concreta que este proponga. Además esta herramienta ayuda a los profesores a diseñar las pruebas de evaluación, aportando también la opción de revisar cualitativamente las soluciones aportadas por los estudiantes. La corrección de una solución enviada se realiza mediante la ejecución de varios test de pruebas y según el número de pruebas que haya superado se asigna una puntuación a cada ejercicio. La arquitectura de LEARN-SQL destaca por su flexibilidad en la implementación y distribución geográfica de los componentes. (Casany, 2013)

SQLify (Computer Assisted Assessment of SQL query Skills) es una nueva herramienta integral para la enseñanza y evaluación de las habilidades de escritura de SQL. La misma incluye revisión por pares, ofreciendo a los estudiantes una experiencia de aprendizaje más rica. Además utiliza un método

relativamente complejo para asignar las calificaciones finales a las asignaciones, diseñados para proporcionar una gama mucho más amplia de calificaciones que simplemente correcta o incorrecta, utiliza la teoría de base de datos para llegar a una evaluación asistida por ordenador, lo que arroja un mejor ambiente de aprendizaje, y reducir el número de intervenciones necesarias realizadas por los instructores del curso (Dekeyser, 2007). Permite ejecutar exclusivamente sentencias SELECT y su función es la de auto-evaluación o la realización de exámenes para los estudiantes. (Masó, 2010)

Learning SQL with a computerized Tutor es una herramienta creada para proporcionar un cierto nivel de asesoramiento y guía mediante el uso de la evaluación y orientación personalizada durante la enseñanza del lenguaje estructurado de consulta. Fue creada con el propósito de explorar y ampliar el modelado basado en restricciones así como la resolución de problemas destinados a complementar la enseñanza en clases. Al igual que SQL-Tutor permite solamente trabajar con sentencias SELECT (Mitrovic, 1998).

1.2.2 Soluciones similares existentes a nivel nacional

Cuba también ha sido partícipe del avance tecnológico que el mundo ha experimentado desarrollando de esta manera sistemas informáticos para contribuir al aprendizaje, un ejemplo significativo de ello es el juez en línea de bases de datos desarrollado en la UCI:

Juez en Línea de Bases de Datos *(Data Base Online Judge)* pensado con el objetivo de procesar y evaluar el código programado en lenguaje de Base Datos para apoyar a los estudiantes y profesores de las asignaturas de Base de Datos I y II (Rodés, 2013). Con el desarrollo del mismo se esperaba acceder a los ejercicios y concursos sobre programación en Base de Datos Relacionales y permitir además que los usuarios, pudieran entrenar y comprobar sus conocimientos en materia de programación de sentencias SQL en el lenguaje PL/pgSQL. A pesar de la gran importancia que se le concede a esta aplicación, no se posee evidencias que demuestren su existencia.

Los autores consideran luego de un análisis exhaustivo de las diferentes herramientas para el apoyo al aprendizaje de SQL mencionadas anteriormente, que aunque todas ellas evalúan de forma automática la solución enviada por el estudiante y en la mayoría de los casos ofrecen la corrección de las mismas; solamente se centran en el trabajo con el tipo más básico de ejercicio SQL: las sentencias SELECT. Además cada una de ellas se ha desarrollado para un entorno y contexto determinado, con sus particularidades específicas. Por otra parte, de acuerdo a las bibliografías consultadas las herramientas antes mencionadas poseen código propietario.

Actualmente existe una amplia variedad de tecnologías y herramientas que permiten la creación y el desarrollo de aplicaciones web.

1.3 Ambiente de desarrollo

El desarrollo de un software es un proceso complejo, en el cual se tienen muchos elementos que permitirán su construcción con calidad. Para la implementación de una aplicación, es necesario tener en cuenta diversos aspectos de gran importancia tales como: la metodología, las tecnologías, los lenguajes de programación y modelado así como los distintos servidores sobre los cuales se va a desarrollar el sistema pues de estas dependerá totalmente el éxito en el desarrollo de la misma.

1.3.1 Metodología de Desarrollo

Según la Real Academia Española una metodología de desarrollo de software no es más que un "*conjunto de métodos que se siguen en una investigación científica o en una exposición doctrina*l" (RAE, 2012). Las metodologías imponen un proceso disciplinado sobre el desarrollo de software, ejemplo de ellas son las propuestas tradicionales, centradas específicamente en el control del proceso y las ágiles, que cambian significativamente algunos de los énfasis de los métodos ingenieriles. A pesar de que existen otras metodologías como SCRUM y XP, el equipo de desarrollo analiza la metodología de desarrollo de software tradicional Rational Unified Process (RUP).

RUP está pensado para adaptarse a cualquier proyecto. Unifica completamente al equipo de desarrollo de software y optimiza la productividad de cada uno de los miembros del equipo. Es una metodología dirigida por Casos de Uso, centrada en la arquitectura e iterativa e incremental (Fernandez, 2010). Además utiliza Unified Modeling Language (UML) como lenguaje de notación. El desarrollo de software está dividido por RUP en 4 etapas: Inicio, Elaboración, Construcción y Transición.

La selección de RUP como metodología de desarrollo estuvo determinada por las diversas características para el desarrollo de cualquier proyecto que la misma presenta, ejemplo de ello es que permite unificar completamente al equipo de desarrollo de software, es dirigida por Casos de Uso, centrada en la arquitectura y utiliza UML como lenguaje de notación. Además constituye un proceso iterativo e incremental.

Para desarrollar el sistema informático que se propone, se hace necesario seleccionar los lenguajes tanto de programación como de modelado, las tecnologías y las herramientas a utilizar. A continuación se procede a la selección de las mismas.

En el mundo de desarrollo de aplicaciones informáticas el uso de los frameworks se vuelve inminente, ya que estas estructuras constituyen la base con la cual un proyecto de software puede ser fácilmente desarrollado. (Eguiluz, 2011)

1.3.2 Framework de Desarrollo

Symfony2 ha sido ideado para aprovechar al máximo todas las nuevas características de PHP y por eso es uno de los marcos de trabajo PHP con mejor rendimiento. Es un marco de trabajo rápido, flexible y fácil de aprender que le permite a los desarrolladores construir aplicaciones webs más mantenibles. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no se ajustan a un proyecto. (Eguiluz, 2011)

Symfony 2 separa la lógica de negocio, la del servidor y la presentación de la aplicación. Es un marco de trabajo de código abierto que está construido utilizando un contenedor de inyección de dependencias. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. Por otra parte, es compatible con la mayoría de gestores de bases de datos. (Eguiluz, 2011)

La selección se basó en el conocimiento que se posee sobre el framework Symfony 2, evitando invertir tiempo y esfuerzo en el aprendizaje de una nueva tecnología de este tipo y aprovechar el mismo en el desarrollo de la propuesta de solución. También se selecciona gracias a la peculiaridad que posee de tener su arquitectura interna totalmente desacoplada, lo cual le permite adaptarse con facilidad a los nuevos cambios que puedan surgir durante el proceso de desarrollo del software. También brinda la facilidad de utilizar el ORM y Doctrine con lo que se logra la independencia del gestor de base de datos y hace uso de HTML 5 y CSS 3. La versión a utilizar será la 2.5.1.

A continuación se hace alusión a los lenguajes utilizados no solo para ser interpretados por el navegador de los clientes, sino también en el modelado de los artefactos del sistema.

1.3.3 Lenguajes del lado del servidor

PHP: Es un lenguaje de código abierto, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. PHP puede hacer cualquier tarea que se pueda realizar con un script como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies (Mehd ACHOUR, 2006). La selección de Symfony como framework de desarrollo en el servidor, determina la utilización de PHP para desarrollar la propuesta de solución, debido a que Symfony está escrito en dicho lenguaje. La versión a utilizar es la 5.4.12.

1.3.4 Lenguajes del lado del cliente

JavaScript: Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. Es un lenguaje sencillo y muy útil. Además, contiene una serie de funciones y sentencias utilizadas para facilitar la interacción con los documentos HTML, poder manipular de manera sencilla el DOM, desarrollar aplicaciones usando AJAX[3] y manipular eventos (Pérez, 2009). La versión a utilizar es la 1.8.

HTML: Es un lenguaje comúnmente utilizado para la publicación de hipertexto[4] en la Web. HTML utiliza etiquetas que marcan elementos y estructuran el texto de un documento. La última versión de este lenguaje es la conocida HTML5, con la misma HTML provee los elementos estructurales, CSS[5] se encarga de volver esa estructura utilizable y atractiva a la vista, y JavaScript tendrá el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales y con mayor capacidad de interacción con el usuario (GAUCHAT, 2012). Debido a las novedosas herramientas que incorpora para el desarrollo web, se considera la selección de HTML5 como el lenguaje para realizar el maquetado de la propuesta de solución.

CSS: Las hojas de estilo son complementos de código añadidos al HTML que se encargan de la apariencia debido a que son las más conocidas y utilizadas para definir las propiedades de formato de los diferentes elementos HTML (LANCKER, 2009). La selección de CSS como lenguaje para escribir el estilo visual de la propuesta de solución se hizo teniendo en cuenta que el mismo posee un amplio uso por parte de maquetadores, diseñadores y desarrolladores web. La versión a utilizar será la 3.0.

JQuery: es un framework JavaScript, implementa una serie de clases (POO) que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM[6], manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web. Es un producto bien documentado y con un gran equipo de desarrolladores a cargo de la mejora y actualización del framework (Ojeda, 2013). La versión a utilizar es la 1.10.2.

Bootstrap: es un framework muy útil en el desarrollo de sitios web debido a que el mismo está basado en los últimos estándares de desarrollo Web HTML5, CSS3 y JavaScript/JQuery, además es compatible con todos los navegadores habituales (Mestras, 2013). Bootstrap 3 añade nuevos elementos y cambia algunos

[3] **AJAX:** Asynchronous JavaScript + XML
[4] **Hipertexto:** documento que contiene información cruzada con otros documentos, lo cual permite pasar de un documento al referenciado desde la misma aplicación que se está visualizando.
[5] **CSS**: Cascading Style Sheet.
[6] **DOM:** Document Object Manager.

de los ya existentes, ejemplo de ello son: .glyphicon, .panel, .panel-default, .panel-body, .panel-title, .panel-heading, .panel-footer, .panel-collapse etc. (Thornton, 2014). Por otra parte todos los plugins JavaScript de Bootstrap requieren la librería jQuery para funcionar. La selección de Bootstrap se basa teniendo en cuenta que el mismo es adaptable a las resoluciones de pantalla, es ágil en la construcción de interfaces y contiene gran cantidad de componentes. La versión a utilizar será la 3.0.

1.3.5 Tecnología del lado del Cliente.

AJAX: Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes. Las tecnologías que forman AJAX definidas en (Holdener, 2008) son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano. (Euguiluz, 2005)

1.3.6 Lenguaje de Modelado

El Lenguaje de Modelado (UML) "*es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software*" (Jacobson, 2000). UML cuenta con un conjunto de notaciones y diagramas para modelar sistemas orientados a objetos y puede usarse en las diferentes etapas del ciclo de vida del desarrollo del software. La especificación de UML no define un proceso estándar pero está ideado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Jacobson, 2000). La capacidad de UML para modelar cualquier sistema de software y el hecho de que haya sido adoptado por el OMG[7] como un estándar desde Noviembre de 1997, permitieron determinarlo como el lenguaje de modelado para desarrollar la propuesta de solución. La versión a utilizar es la 2.1.

7**OMG**: Grupo de Gestión de Objetos

1.3.7 Herramienta CASE para el Modelado UML

Visual Paradigm for UML: Es una herramienta que soporta el modelado mediante UML y proporciona asistencia a los analistas y desarrolladores durante todas las etapas del ciclo de vida de desarrollo de un software. Permite diseñar todo tipo de diagrama de clases y generar documentación (GAMMA, 2005). Por otra parte, apoya al ciclo de vida completo de desarrollo de software en diversos IDE como NetBeans (Puente, 2011). La versión a utilizar para el desarrollo de la propuesta de solución será la 8.0.

1.3.8 Servidor de Base de Datos

PostgreSQL: Es un SGBD[8] más antiguos y conocidos. Clasifica como software libre por lo que su código fuente puede ser utilizado, modificado, redistribuido y hasta incluido en software con carácter no libre (Martinez, 2014). Funciona muy bien con grandes cantidades de datos y dentro de sus características más relevantes se pueden destacar que es multiplataforma puesto que está disponible para plataformas Linux, Unix y Windows. Permite realizar copias de seguridad y múltiples métodos de autenticación y acceso encriptado, así como la replicación sincrónica y asincrónica. Además posee numerosos tipos de datos y ofrece la posibilidad de definir nuevos. La versión a utilizar será la 9.2.4. Para su selección como el gestor de bases de datos para el desarrollo de la propuesta de solución se tuvo en cuenta su carácter libre, el conocimiento previo sobre el mismo y la facilidad de manejo de las bases de datos PostgreSQL.

1.3.9 Entorno de Desarrollo Integrado

NetBeans IDE: Es un proyecto de código abierto dedicado a proveer un sólido desarrollo de software, dirigido fundamentalmente a las necesidades de los desarrolladores y los usuarios; dotándolos de una herramienta para el desarrollo rápido y fácil de productos de software (Community, 2013). NetBeans es una herramienta modular de desarrollo que incluye un avanzado editor multilenguaje y un detector de errores, además permite la integración con PHP, la librería JQuery de JavaScript y Symfony 2. Además cuenta con una amplia documentación y una gran comunidad de usuarios. (Community, 2013) La versión a utilizar es la 7.3.

1.3.10 Servidor Web

Apache: Es un servidor web HTTP de código abierto. Combina escalabilidad, seguridad y rendimiento. Es altamente configurable de diseño modular, gratuito, trabaja con Perl, PHP y otros lenguajes de Script y funciona en Windows y en Linux, así como en otros sistemas Unix (Kabir, 2003). La selección de Apache

[8]**SGBD:** Sistemas Gestores de Base de Datos.

como servidor web estuvo determinada por la utilización de XAMPP[9], además debido a que, entre otros motivos, está respaldado por una comunidad de usuarios numerosa, por lo que existe una documentación extensa del mismo disponible en Internet. La versión a utilizar es la 2.4.10.

XAMPP: Es un servidor independiente multiplataforma, de software libre, que consiste principalmente en la base de datos MySQL, el servidor web *Apache* y los intérpretes para lenguajes de script: PHP y Perl. Liberado bajo la licencia GNU y actúa como un servidor web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris y MacOS X. Muy útil para la elaboración de páginas dinámicas, fácil de instalar y las configuraciones son mínimas o inexistentes, lo cual nos ahorra bastante tiempo (Ibarra, 2013). La versión a utilizar es la 3.2.1.

1.3.11 Mapeador de Objetos Relacionales

Doctrine ORM: es un mapeador de objetos-relacional (ORM) para PHP que proporciona una capa de persistencia para objetos PHP. Es una capa de abstracción que se ubica justo encima de un SGBD permitiendo trabajar los objetos de una base de datos relacional con un paradigma diferente al de captar filas de una tabla basada en columnas de una matriz. Una de las características clave de Doctrine es la opción de escribir las consultas de base de datos en un dialecto Structured Query Language (SQL) propio orientado a objetos llamado Doctrine Query Language (DQL). (Pacheco, 2011)

1.4 Conclusiones Parciales

La identificación de los aspectos teóricos como SQL, Jueces en línea y los Sistemas Tutores Inteligentes fundamentan la necesidad de desarrollar un ambiente inteligente para el aprendizaje del lenguaje estructurado de consultas que garantice brindar soporte a los estudiantes. El estudio realizado para conformar el ambiente de desarrollo permitió seleccionar a RUP como metodología de desarrollo de software, UML 2.1 como lenguaje de modelado, Visual Paradigm 8.0 como herramienta CASE de modelado, los framework Bootstrap 3.0, JQuery 1.10.2 y Symfony 2. Además permitió seleccionar a Doctrine ORM como mapeador de objetos relacionales, NetBeans 7.3 como IDE, Apache 2.4.10 como servidor web, PostgreSQL 9.2.4 como servidor de Base de Datos, PHP 5.4.12 como lenguaje del lado del servidor y CSS 3, JavaScript 1.8, Ajax y HTML 5 como lenguajes del lado del cliente.

[9] **XAMPP:** paquete formado por un servidor web Apache, una base de datos MySQL y los intérpretes para los lenguajes PHP y Perl.

CAPÍTULO II

Capítulo 2: Propuesta de solución

Introducción

En este capítulo se realiza una descripción de la solución propuesta con el objetivo de proporcionar un mejor entendimiento del sistema. Además se representa el diagrama del Modelo del Dominio con la correspondiente descripción de cada uno de sus elementos. Por otra parte se especifican los requisitos funcionales y no funcionales que debe cumplir el sistema. Asimismo, se obtienen como artefactos fundamentales los casos de uso arquitectónicamente significativos y sus descripciones.

2.1 Descripción del Sistema

La solución que se propone tiene como objetivo apoyar el aprendizaje del lenguaje estructurado de consulta en la asignatura Sistema de Bases de Datos. El sistema a implementar consiste en el desarrollo de un ambiente inteligente, que permita realizar la corrección de los ejercicios realizados por los estudiantes y a la vez brindar información sobre los errores cometidos.

La herramienta a desarrollar cuenta con una sesión destinada a los **administradores** que serán los encargados además de otras acciones, de gestionar los usuarios y los problemas. A cada problema que se gestione en el sistema, los administradores deberán incluirle una solución para que el mismo pueda publicarse online, de lo contrario permanecerá offline y los estudiantes no podrán ofrecerle solución. La solución propuesta por los administradores será utilizada posteriormente por el sistema con el objetivo de evaluar la respuesta al problema formulada por el estudiante.

La aplicación contará además con una sesión destinada a los **estudiantes** que serán los encargados de dar respuesta a los problemas publicados y tendrán además la posibilidad de desarrollar otras acciones.

Para desarrollar la evaluación el estudiante debe haber enviado una solución a un problema, el **Sistema** se encargará de la corrección automática de la misma ejecutando la solución del autor y la solución propuesta por el estudiante, confrontando de esta manera los resultados y comprobando si los mismos coinciden. Una vez comprobados los resultados el sistema envía una calificación, la misma puede ser Aceptado, Respuesta Incorrecta o Error de Compilación.

Luego de enviada la evaluación, si la misma es *Error de Compilación* el estudiante tendrá la opción de obtener información específica de los errores cometidos.

Para generar la retroalimentación el sistema verificará la solución del estudiante teniendo en cuenta un conjunto de restricciones del lenguaje. Las mismas, están formadas teniendo en cuenta una condición de relevancia (Cr) y una condición de satisfacción (Cs) de la siguiente manera:

Si se cumple <Cr> entonces <Cs> debe cumplirse también.

La solución del estudiante se verficará teniendo en cuenta la condición planteada anteriormente y se emitrá la retroalimentación teniendo en cuenta el error cometido por el estudiante.

Con esta aplicación el alumno dispondrá de un tutor disponible a todas horas, que le permite saber el estado de la resolución de sus actividades y, en caso de error, recibir las indicaciones que le guíen hacia la solución correcta.

2.2 Modelo de Dominio

El modelo de dominio es utilizado por el analista como un medio para comprender el sistema debido a que es una especie de representación esquemática de los conceptos y elementos de la vida real que serán usados en el mismo. Se crea con el fin de representar los conceptos claves del dominio del problema, las propiedades más importantes y las relaciones entre los conceptos facilitando una mejor comunicación entre desarrolladores y clientes al establecer un lenguaje común para el entendimiento del mismo. (Hans-Erik Eriksson, Magnus Penker, 2000)

2.2.1 Diagrama de clases del Modelo de Dominio

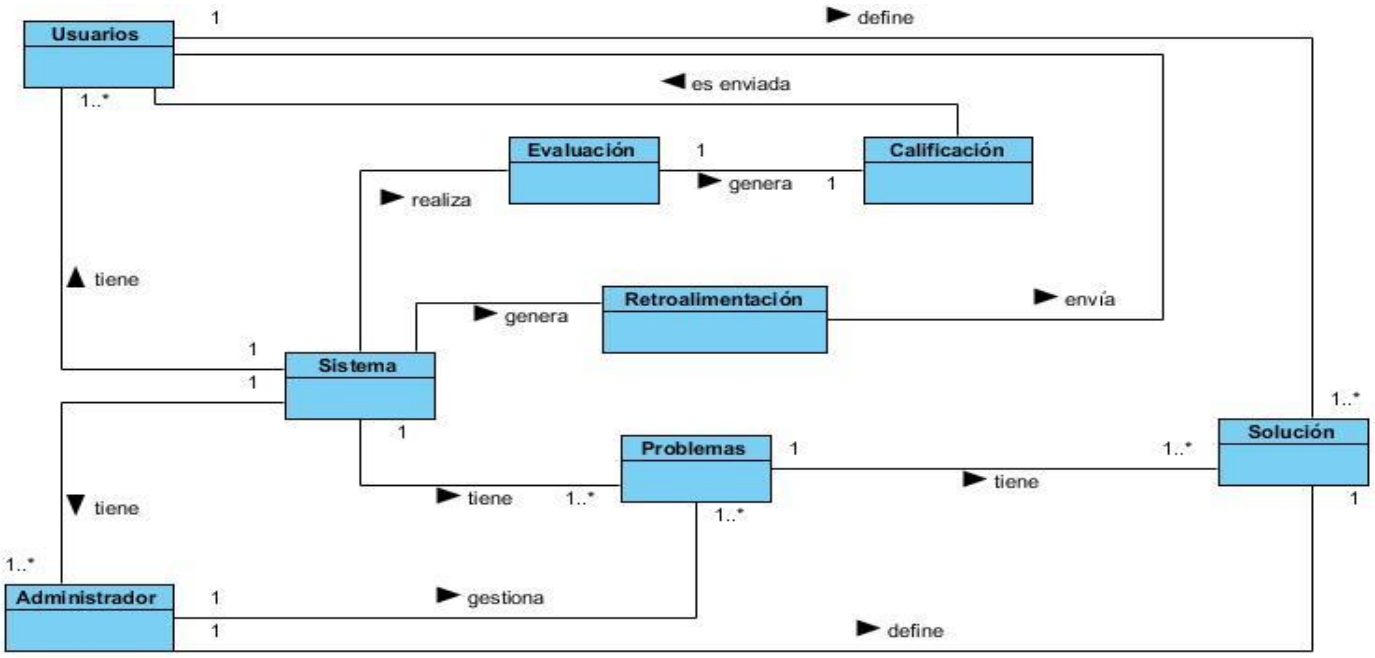


Figura 4 Diagrama de clases del Modelo de Dominio

2.2.2 Definición de las clases del Modelo de Dominio

A continuación se muestra la definición de cada una de las clases del modelo de dominio:

Usuario: Identifica a los estudiantes que tendrán acceso al sistema con el objetivo de interactuar con el mismo para dar solución a los problemas, recibir evaluación, retroalimentación y realizar otras acciones.

Administrador: Identifica al usuario que tendrá acceso al sistema con el objetivo de gestionar a los usuarios, los problemas y llevar a cabo otras acciones.

Sistema: Identifica a la aplicación que contiene los problemas, se encargará de realizar la evaluación de las soluciones y a su vez generar la retroalimentación teniendo en cuenta la solución enviada por los estudiantes.

Retroalimentación: Información en forma de ayuda que se proporciona al estudiante, teniendo en cuenta la evaluación recibida.

Evaluación: Proceso a través del cual se evalúa la solución enviada por el estudiante.

Calificación: Calificación obtenida por el estudiante una vez realizada la evaluación.

Problemas: Conjunto de ejercicios formulados por los administradores con el objetivo de que los estudiantes le provean respuesta.

Solución: Respuestas a los ejercicios definidas por los estudiantes y administradores.

2.3 Especificación de requisitos

El levantamiento de los requisitos constituye una de las etapas más importantes en el proceso de ingeniería de requisitos. Para logar el éxito durante el desarrollo de esta etapa, se realiza un trabajo en conjunto entre los participantes y los desarrolladores con el objetivo de identificar el problema, proponer elementos de solución, negociar diferentes enfoques y especificar un conjunto preliminar de requisitos para la solución. (Pressman, 2005)

2.3.1 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera que éste debe reaccionar a entradas particulares y cómo se debe comportar en situaciones particulares (Sommerville, 2005). Seguidamente se muestran los requisitos funcionales definidos para la siguiente investigación:

RF1 Ver Problemas: Permite al invitado visualizar los datos de los problemas existentes en el sistema sin necesidad de autenticación.

RF2 Filtrar Problemas: Permite buscar un problema específico en el sistema.

RF3 Ver ranking: Permite visualizar en un listado, el lugar que ocupan todos los usuarios teniendo en cuenta el acumulado de puntos obtenidos con la evaluación de las soluciones.

RF4 Ver envíos: Permite visualizar todos los envíos realizados a un problema determinado. Cada envío, corresponde a una solución enviada por un estudiante a dicho problema.

RF5 Buscar Usuario: Permite buscar un usuario específico dentro del sistema.

RF6 Autenticar usuario: Permite autenticarse en el sistema utilizando usuario y contraseña. Una vez autenticado, el usuario tiene permiso de realizar las acciones que le sean permitidas teniendo en cuenta el rol que posea.

RF7 Ver Perfil: Permite al usuario del sistema visualizar los datos de su perfil.

RF8 Configurar Perfil: Permite al usuario del sistema editar su cuenta de usuario.

RF9 Registrar Usuario: Permite a cualquier persona registrarse en el sistema con el objetivo de poder autenticarse y de esta forma poder interactuar con el mismo.

RF10 Ver mis envíos: Permite visualizar los envíos a un problema teniendo en cuenta un usuario determinado.

RF11 Enviar Solución: Permite al estudiante proporcionar una solución a un problema, la misma una vez insertada contará como un envío.

RF12 Evaluar solución: Permite realizar de forma automática la evaluación de una solución a un problema proporcionada por un estudiante.

RF13 Generar retroalimentación: Permite generar la retroalimentación teniendo en cuenta los errores cometidos por el estudiante en la solución enviada, de esta manera se le entregará al estudiante información específica de los errores cometidos.

RF14 Incluir Solución de Autor: Permite al administrador insertar una solución a un problema. La misma será utilizada por el sistema para evaluar la solución enviada por el estudiante a ese problema.

RF15 Editar Solución de Autor: Permite al administrador editar la solución a un problema que haya sido creada anteriormente.

RF16 Ver solución de estudiante: Permite al administrador visualizar la solución a un problema enviada por un estudiante.

Gestionar Usuarios

RF17 Insertar usuario: Permite al administrador insertar un usuario en el sistema, los mismos se irán almacenando en un listado.

RF18 Editar usuario: Permite al administrador modificar los datos de un usuario creado anteriormente.

RF19 Mostrar usuario: Permite al administrador visualizar los datos de un usuario creado anteriormente.

RF20 Eliminar usuario: Permite al administrador eliminar un usuario creado anteriormente.

Gestionar Problemas

RF21 Insertar problema: Permite al administrador insertar un problema en el sistema, los mismos se irán almacenando en un listado de problemas.

RF22 Editar problema: Permite al administrador editar los datos de un problema.

RF23 Mostrar problema: Permite al administrador visualizar los datos de un problema.

RF24 Eliminar problema: Permite al administrador eliminar un problema del sistema.

RF25 Ver calificación: Permite al estudiante observar la calificación de las soluciones enviadas.

RF26 Ver retroalimentación: Permite al estudiante observar la ayuda que proporciona el sistema.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales, como su nombre sugiere, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las restricciones de los servicios o funciones ofrecidas por el sistema. Normalmente solo se aplican a características o servicios individuales del sistema (Sommerville, 2005). A continuación se muestran los requisitos no funcionales definidos para la investigación:

- ✓ **Confiabilidad:**

RNF1: Solo tendrán acceso al sistema los usuarios registrados con una contraseña y usuario válido.

RFN2: El sistema debe ofrecer una advertencia antes de realizar acciones irreversibles (Ejemplo: borrar cualquier dato).

RFN3: Seguridad de acceso y administración de usuarios: otorgamiento roles. Los niveles de acceso están determinados por los diferentes roles válidos dentro del sistema.

✓ **Usabilidad:**

RNF4: El desarrollo del sistema tiene como propósito evaluar las soluciones a los problemas enviadas por los estudiantes y a su vez brindar retroalimentación teniendo en cuenta los errores cometidos.

✓ **Restricciones de diseño e implementación:**

RNF5: Para el análisis y el diseño del sistema debe ser empleado el lenguaje de modelado UML y como herramienta CASE para el modelado Visual Paradigm en su versión 8.0.

RNF6: El sistema debe ser implementado haciendo uso del IDE NetBeans en su versión 7.3.

RNF7: El marco de trabajo que se debe utilizar para el desarrollo es el framework Symfony en su versión 2.5.1.

RNF8: Se debe utilizar para el desarrollo, como lenguaje de programación del lado del servidor PHP en su versión 5.4.12.

RNF9: Se debe utilizar para el desarrollo del lado del cliente HTML en su versión 5, CSS 3 y como lenguaje de scripting debe ser empleado Java Script haciendo uso de la librería JQuery.

RNF10: Se debe utilizar como gestor de base de datos PostgreSQL en su versión 9.2.4 y como servidor web Apache en su versión 2.4.10.

✓ **Interfaz:**

RNF11: La interfaz del sistema debe ser de fácil comprensión.

RNF12: La interfaz del sistema debe contar con una buena organización permitiendo de esta manera un mejor manejo del mismo por parte de los usuarios.

RNF13: El sistema debe ser capaz de notificar a los usuarios de la presencia de un error.

✓ **Potenciales:**

RNF14: Las pruebas al sistema deben identificar errores que puedan convertirse en defectos.

2.4 Modelo de Casos de Uso del Sistema

El modelo de casos de uso (CU) constituye el conjunto de todos los actores y los CU que describen las diferentes formas de interactuar con el sistema. Un CU es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para un actor. (Jacobson, 2000)

Seguidamente se muestran los CU identificados a partir de los requisitos funcionales descritos anteriormente:

1. Autenticar Usuarios.
2. Ver Perfil.
3. Configurar Perfil.
4. Registrar Usuarios.
5. Ver mis Envíos.
6. Enviar Solución.
7. Ver problema.
8. Filtrar Problemas.
9. Buscar Usuarios.
10. Ver Estadísticas.
11. Ver Envíos.
12. Gestionar Usuario.
13. Gestionar Problemas.
14. Incluir Solución de Autor.
15. Editar Solución de Autor.
16. Ver Solución de Estudiantes.
17. Evaluar Solución.
18. Generar Retroalimentación.
19. Ver Calificación.
20. Ver Retroalimentación.

2.4.1 Descripción de los actores

Los actores identificados son las personas o sistemas encargadas de inicializar e interactuar con los casos de uso del sistema. A continuación se describen los actores del sistema:

Actores	Descripción
Administrador	Actor que identifica a los usuarios que interactúan con el sistema con el

	objetivo de gestionar los usuarios que tendrán acceso al mismo así como los problemas que en este se incluyan. Se encargará además de suministrarle solución a los problemas creados la cual será utilizada por el sistema para evaluar la solución enviada por los estudiantes. Además tiene la posibilidad de realizar las mismas acciones que realiza el usuario del sistema.
Estudiantes	Actor que identifica a todos los usuarios que interactúan con el sistema con el objetivo de solucionar los problemas, recibir evaluación y retroalimentación. Además tiene la posibilidad realizar las mismas acciones que el usuario del sistema.
Invitado	Actor que interactúa con el sistema con el objetivo de realizar una serie de acciones sin necesidad de autenticación (ver los problemas, ver los envíos, ver las estadísticas, filtrar problemas, buscar usuarios).
Usuario del Sistema	Actor que identifica a los usuarios que se autentican para interactuar con el sistema (administrador y estudiantes). Además puede realizar una serie de acciones tales como configurar perfil y ver el perfil.
Sistema	Actor que permite que se realice la evaluación y la retroalimentación de forma automática e invisible a los usuarios.

Tabla 1 Descripción de los actores del Sistema.

2.4.2 Patrones de Casos de Uso

Los patrones de casos de uso son comportamientos que deben existir en el sistema, describen el uso del mismo y cómo este interactúa con los usuarios. Durante la elaboración del diagrama de CU se tuvo en cuenta el uso de patrones. La aplicación de los mismos permite una descripción más detallada de la interacción entre usuarios y el sistema y facilita un mejor entendimiento con los clientes.

A continuación se muestran los patrones aplicados:

CRUD (Creating, Reading, Updating, Deleting) constituye un ejemplo del uso de patrones en la investigación. CRUD modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación.

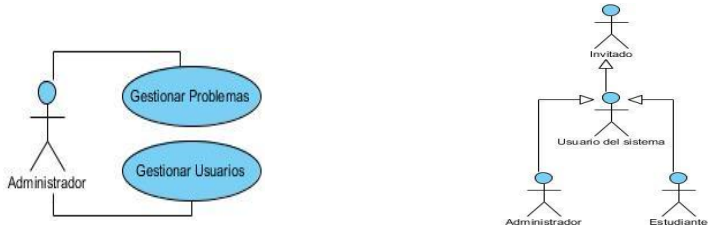


Figura 5 Ejemplo de Patrones de CU (CRUD, Múltiples Actores y Roles Comunes)

Múltiples actores y Roles común en el caso en que uno o más actores juegan el mismo rol sobre un caso de uso, representándose por un actor, heredado a su vez por otros que comparten funcionalidades.

2.4.3 Diagrama de casos de uso del sistema

El diagrama de Casos de Uso (DCU) muestra un conjunto de actores y CU con una asociación entre cada par actor/CU que representa gráficamente las funcionalidades principales del sistema y su interacción con los actores. (Jacobson, 2000)

A continuación se muestra el DCU del sistema definido para la presente investigación:

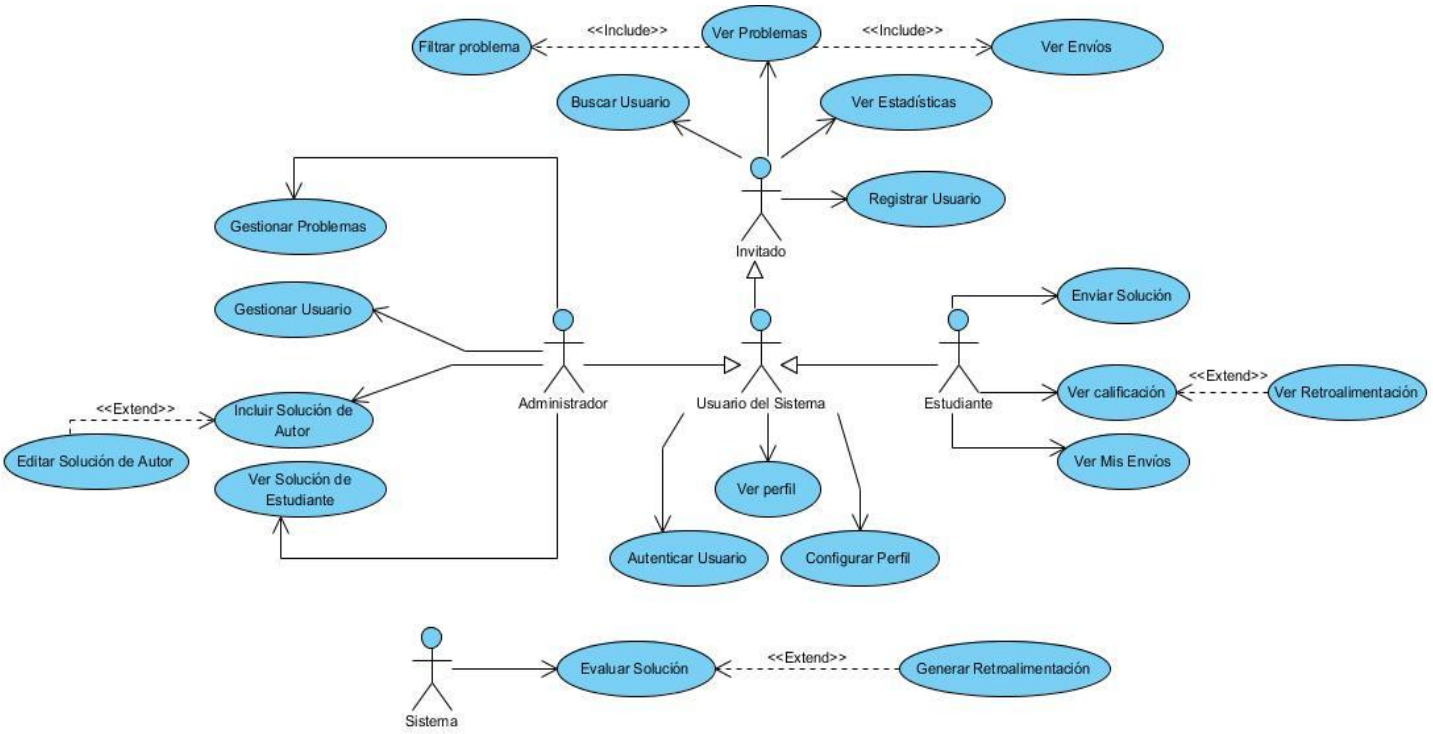


Figura 6 Diagrama de CU del sistema AIA-SQL

2.4.4 Descripción de casos de uso

A continuación se presenta la descripción del CU Incluir Solución de Autor, los restantes serán mostrados en el Anexo #2 de la presente investigación.

CU- Incluir solución de autor		
Objetivo	Incluir una solución correcta a un problema.	
Actores	Administrador	
Resumen	El caso de uso inicia cuando un administrador crea un problema, para que este sea puesto *online* debe poseer una solución que será introducida por el administrador. Esta solución luego será usada como guía en la evaluación de la solución del estudiante.	
Complejidad	Baja	
Prioridad	Media	
Precondiciones	El administrador debe haber creado un problema. El sistema se encuentra en la interfaz Mostrar Problema.	
Postcondiciones	El administrador incluye una solución a un problema.	
Flujo de eventos		
Flujo básico <Incluir Solución del Estudiante>		
	Actor	**Sistema**
1	Selecciona la opción **Solución Autor.**	
2		Muestra una ventana donde el administrador puede introducir la solución al problema.
3	Inserta la solución y selecciona la opción **Enviar**.	
4		Muestra información sobre el resultado de ejecutar la solución y permite: Editar la solución. Limpiar datos.
6		Coloca el problema *online* en la lista

		de problemas.
7		Termina el Caso de Uso.
Flujos alternos		
El estudiante selecciona la opción **Portada** que le permite regresar a la portada.		
	Actor	**Sistema**
1		Regresa a la portada.
El estudiante selecciona la opción **Limpiar**		
	Actor	**Sistema**
1		Elimina todos los valores asignados a los campos que ya han sido completados.
2		Regresa al paso 2 del flujo básico.
	El campo solución está vacío.	
	Actor	**Sistema**
1		Muestra un mensaje de información.
2		Regresa al paso 2 del flujo básico.
Referencia a los requisitos	RF14	

Tabla 2 Descripción CU Incluir Solución de Autor.

2.5 Conclusiones Parciales

Con el desarrollo de este capítulo se ha concretado la propuesta de solución que servirá como punto de partida para el flujo de trabajo de Análisis y Diseño. Para ello quedaron definidos los principales conceptos, los cuales se relacionaron en el modelo de dominio, el cual ayudó a comprender el entorno en que se relacionan las funcionalidades. Por su parte los requisitos funcionales y no funcionales permitieron definir las capacidades y las cualidades que deben tener las funcionalidades correspondientes al sistema. También se obtuvo a través del diagrama de casos de usos del sistema las relaciones que se establecen entre actores y casos de uso. Finalmente, se describieron los casos de uso para un mejor entendimiento del sistema.

CAPÍTULO III

Capítulo 3: Análisis y Diseño

Introducción

En este capítulo se realiza el análisis y diseño de la propuesta de solución planteada debido a que la metodología de desarrollo RUP propone como tercer flujo de trabajo el análisis y diseño. Se muestra el modelo de análisis y el modelo de diseño generando los artefactos correspondientes en cada uno de ellos. Asimismo se define el patrón arquitectónico y los patrones de diseño que sustentarán todo el proceso de desarrollo. Por otra parte se diseña el modelo de Bases de Datos y el modelo de despliegue.

3.1 Modelo de análisis

Con el desarrollo del modelo de análisis se persigue transformar el modelo de CU en un modelo del diseño, es decir, en una estructura de clasificadores y realizaciones de CU. Además este tiene como objetivo realizar los CU de una forma económica de manera que el sistema ofrezca un rendimiento adecuado y pueda evolucionar en el futuro (Jacobson, 2000). Un modelo de análisis se describe utilizando el lenguaje de los desarrolladores y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los aspectos internos del sistema. Es utilizado fundamentalmente por los desarrolladores para comprender como debería ser diseñado e implementado el sistema. (Jacobson, 2000)

3.1.1 Diagrama de clases del análisis (DCA)

Una clase de análisis representa una abstracción de una o varias clases, ajustando las mismas a uno de los tres estereotipos existentes sobre las clases utilizados por el modelo de dominio: de interfaz, de control o de entidad.

La clase ***Interfaz*** se utiliza generalmente para modelar la interacción entre el sistema y los actores, la clase de ***Control*** es utilizada habitualmente para representar coordinación, secuenciación, transacciones y son las encargadas de manejar y coordinar las acciones y los flujos de control principal, por su parte la clase ***Entidad*** es usada para modelar la información que tiene una vida larga y a veces es persistente, asimismo muestran una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema. (Jacobson, 2000)

A continuación se muestra el diagrama de clases del análisis correspondiente al CU Incluir Solución de Autor. Los diagramas de clases del análisis correspondiente a los restantes CU se muestran en el Anexo #3.

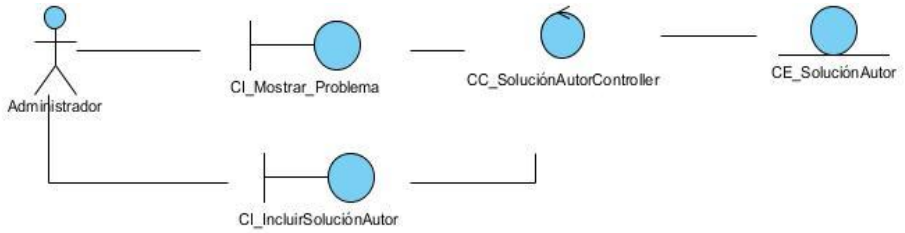


Figura 7 DCA-Incluir Solución de Autor.

3.1.2 Diagrama de colaboración del análisis (DCO)

Los diagramas de colaboración del análisis son utilizados fundamentalmente para modelar las interacciones entre los objetos en el análisis. Estos recuerdan los diagramas de clases pero contienen instancias y enlaces en lugar de clases y asociaciones, mostrando cómo interactúan los objetos secuencialmente o en paralelo enumerando los mensajes que se envían unos a otros. (Jacobson, 2000)

A continuación se muestran el diagrama de colaboración correspondiente al CU Incluir Solución de Autor. Los diagramas de colaboración correspondiente a los restantes CU se muestran en el Anexo #4.

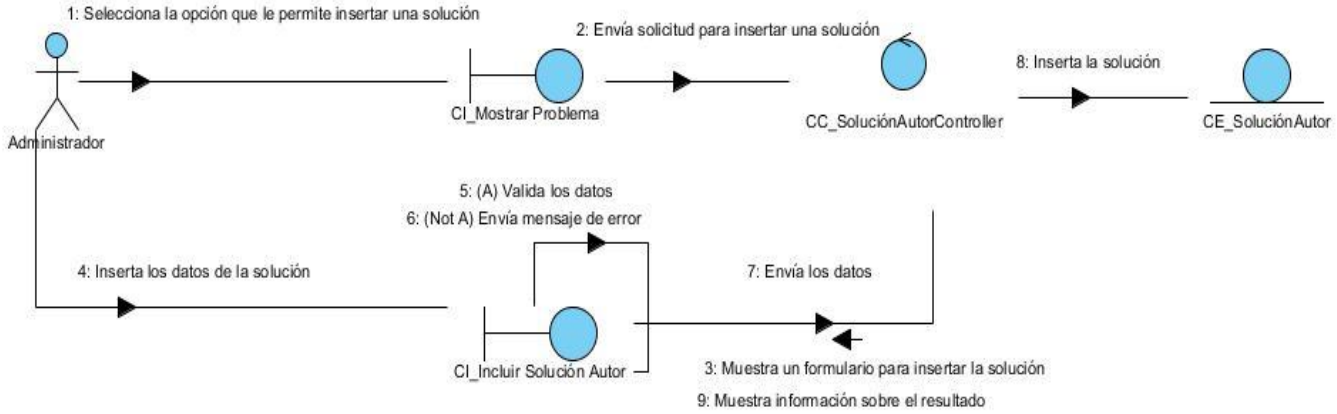


Figura 8 DCO-Incluir Solución de Autor.

3.2 Patrón Arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, son patrones de diseño que ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. (Pressman, 2005)

3.2.1 Patrón Modelo-Vista-Controlador (MVC)

Como base para el desarrollo de la aplicación propuesta se utilizó el marco de trabajo Symfony que implementa, como lo hacen muchos otros, el patrón arquitectónico Modelo-Vista-Controlador (MVC). El patrón MVC separa la lógica de negocio de la interfaz de usuario facilitando de esta forma la evolución por separado de ambos aspectos incrementando la reutilización y la flexibilidad. Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles (Larman, 2003) :

- ***Modelo*** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Está representado por las clases *Entity* que posee Symfony.
- ***Vista*** transforma el modelo en una página web que permite al usuario interactuar con ella. Está representado por las clases *View* que trae Symfony, las cuales poseen la extensión *html.twig*.
- **Controlador** se encarga de procesar las interacciones del usuario procesando toda la información necesaria y realiza los cambios apropiados en el modelo o en la vista. Está representado por las clases *Controller*.

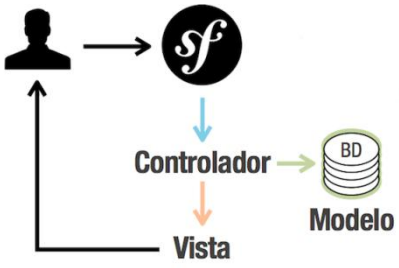


Figura 9 Patón Modelo-Vista-Controlador

El principio más importante de la arquitectura MVC es la separación del código del programa en tres capas, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador siendo esta la principal ventaja de utilizar esta arquitectura. (Larman, 2003)

3.3 Modelo de diseño

El modelo de diseño se crea tomando como base el modelo de análisis para modelar el sistema convirtiéndose en un punto de partida para la implementación. Al igual que el modelo de análisis, este

define clasificadores, relaciones entre estos clasificadores y colaboraciones que llevan a cabo los CU. (Jacobson, 2000)

3.3.1 Patrones de diseño aplicados

Symfony, como framework, hace uso en su implementación de un conjunto de patrones de diseño, los cuales proveen un esquema para refinar los subsistemas y componentes de un sistema de software, o las relaciones entre ellos. “Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software”. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. (Garis, 2012) A continuación se describen los patrones utilizados durante el desarrollo del sistema:

Inyección de Dependencias: Es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. Este patrón es implementado mediante un "contenedor DI"[10]. Dentro del marco de trabajo de Symfony 2 se encuentra la clase *Symfony\Bundle\FrameworkBundle\Controller\Controller* la cual proporciona un atributo público llamado container, una instancia del contenedor de dependencias. Este permite que desde cualquier clase derivada de la mencionada anteriormente se pueda obtener una instancia del contenedor de dependencias, por lo que se puede instanciar cualquier servicio existente haciendo uso de la función *get* (). A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```
public function createAction(Request $request) {

    $entity = new Problema();
    $form = $this->createCreateForm($entity);

    $form->handleRequest($request);


    if ($form->isValid()) {

        $em = $this->getDoctrine()->getManager();
        $entity->setEnvios(0);
        $entity->setAceptados(0);
        $entity->setPorCientoAceptacion(0);
        $entity->setOnline(false);

        //Subir Archivos

        $entity->subirFoto($this->container->getParameter('juez.directorio.problema'));
        $entity->subirArchivoSQL($this->container->getParameter('juez.directorio.archivos'));
```

Figura 10 Ejemplo del uso del patrón Inyección de Dependencias.

[10] Contenedor de Inyección de dependencias.

Symfony es un framework que sigue la mayoría de los patrones de diseño para la web, entre ellos los patrones **GRASP** (Patrones Generales de Software para Asignación de Responsabilidades, traducido al inglés General Responsibility Assignment Software Patterns) y **GoF** (Grupo de 4, traducido al inglés Gang of Four). (Fabien Potencier, 2008)

Patrones GRASP

Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones; son una serie de buenas prácticas enfocadas a la calidad del software (Larman, 2003). A continuación se describen los patrones GRASP utilizados durante el desarrollo:

Experto: Consiste en que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo o ejecutarlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada, es decir, disminuye el acoplamiento (Carmona, 2012). En el sistema desarrollado se aprecia este patrón en las clases entidades, por ejemplo usuario.php es la entidad experta en conocer toda la información referente a los usuarios.

Controlador: Es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Guía la asignación de responsabilidades relacionadas con la creación de objetos, permitiendo instanciar y crear las clases que le son necesarias para cumplir sus funcionalidades. La arquitectura MVC brinda una capa específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón (Carmona, 2012). En la investigación, el uso de este patrón puede evidenciarse en las clases controladoras ubicadas en *Juez/src/JO/AdministracionBundle/Controller.*

Creador: Asigna responsabilidades relacionadas con la creación de objetos o instanciación de nuevos objetos o clases. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (Carmona, 2012). Brinda un soporte a un bajo acoplamiento lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Este patrón es utilizado en las *clases controladoras* en las cuales se crean objetos de las clases entidades para su manipulación.

Alta Cohesión: Brinda una solución al problema de “asignar una responsabilidad de manera que la cohesión permanezca alta” (Larman, 2003). En el caso del sistema en las clases controladoras se evidencia el uso de este patrón pues las mismas están formadas por varias funcionalidades que están

estrechamente relacionadas, siendo las estas las encargadas de definir las acciones y colaborar con otras para realizar tareas que las implican a todas.

Bajo Acoplamiento: Posibilita la idea de tener las clases lo menos relacionadas y en caso de cualquier modificación la repercusión de la misma sea menor potenciando la reutilización (Carmona, 2012). Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, demostrándose así el uso de este patrón. En la solución se puede apreciar que las *clases entidades* son las más reutilizadas.

Patrones GoF

Describen 23 patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño usando modelado UML. Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento. (Larman, 2003)

Los patrones ***creacionales*** abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados. (Abstract Factory, Factory Method, Prototype y Singleton).

Los patrones ***estructurales*** se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. (Adapter, Decorator, Fachada, y Proxy).

Los patrones de ***comportamiento*** están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos. (Chain of Responsability, Interpreter, Observer, Template Method).

A continuación se describen los patrones GoF utilizados durante el desarrollo:

Abstract Factory[11]: es un patrón de tipo creación, su objetivo es facilitar una interfaz para crear "familias" de objetos relacionados o dependientes sin que sea necesario especificar su clase (Rodríguez, 2012). Es utilizado cuando el marco de trabajo necesita crear un nuevo objeto para una petición, el cual busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea, como la definición por defecto de la factoría para las peticiones es *"getRequest ()",* Symfony2 crea un objeto de esta clase para tratar con las peticiones.

[11] **Abstract Factory**: Factoría Abstracta.

Decorator[12]**:** Es un patrón de tipo estructura, ya que permite determinar que clases y objetos serán utilizados para componer estructuras de mayor tamaño. Este patrón añade dinámicamente nuevas responsabilidades a un objeto (SWEAT, 2005). Cada una de las vistas generadas hereda su diseño de la plantilla *"base.html.twig"* pues esta almacena el código HTML que es común a todas las páginas de la aplicación, siendo la plantilla contenedora de la estructura y el diseño básico de las vistas.

3.3.2 Diagrama de clase del diseño (DCD)

Contiene debido a la adaptación del modelo de diseño a la implementación, mayor cantidad de detalles que el diagrama de análisis entre los que podemos citar: clases, atributos, operaciones, subsistemas y relaciones (Jacobson, 2000). A continuación se muestra el diagrama de clase del diseño correspondiente a CU Incluir Solución de Autor. Los restantes diagramas de clase del diseño se muestran en Anexo #5.

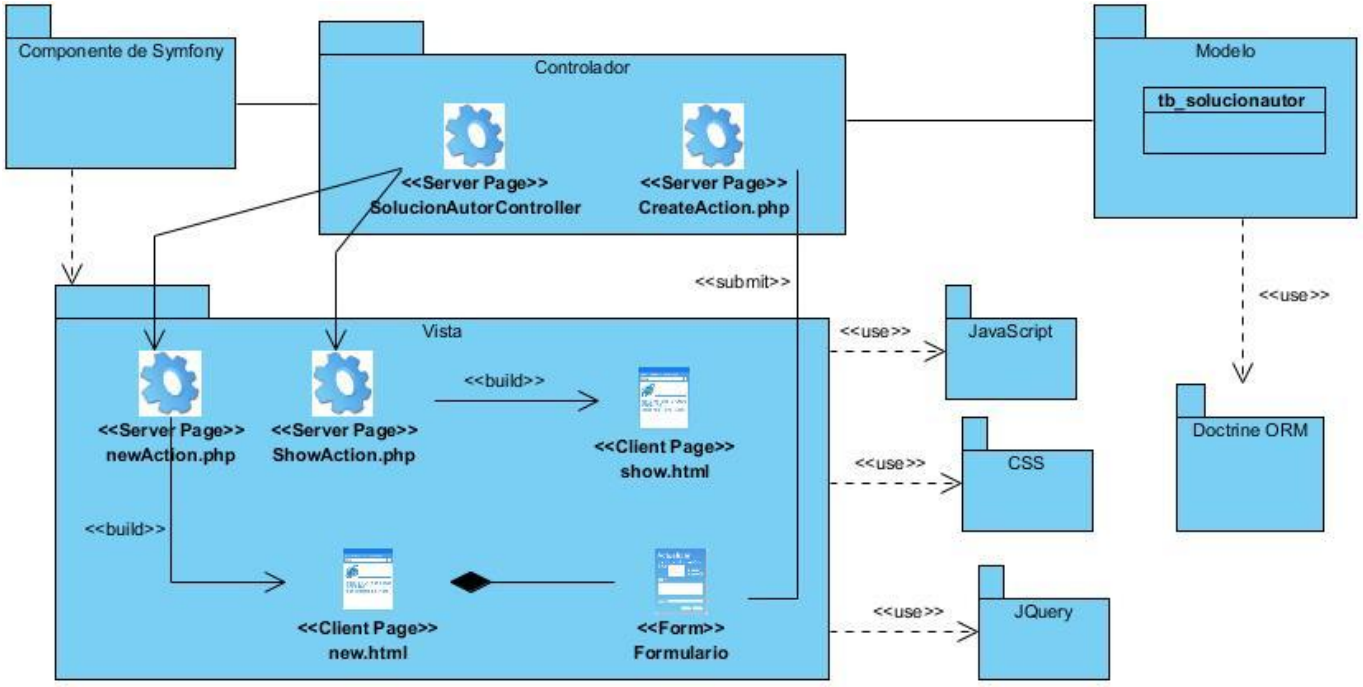


Figura 11 DCD-Incluir Solución de Autor.

3.3.3 Diagrama de secuencia del diseño (DS)

El diagrama de secuencia muestra cómo se pasa de un objeto a otro a medida que se ejecuta el CU relacionándose a través de mensajes. A menudo los desarrolladores utilizan textos para explicar cómo interactúan los objetos de diseño para llevar a cabo el flujo de eventos del CU (Jacobson, 2000). A

[12] **Decorator** Decorador.

continuación se muestra el diagrama de secuencia correspondiente al CU Incluir Solución de Autor. Los restantes diagramas se muestran en el Anexo #6.

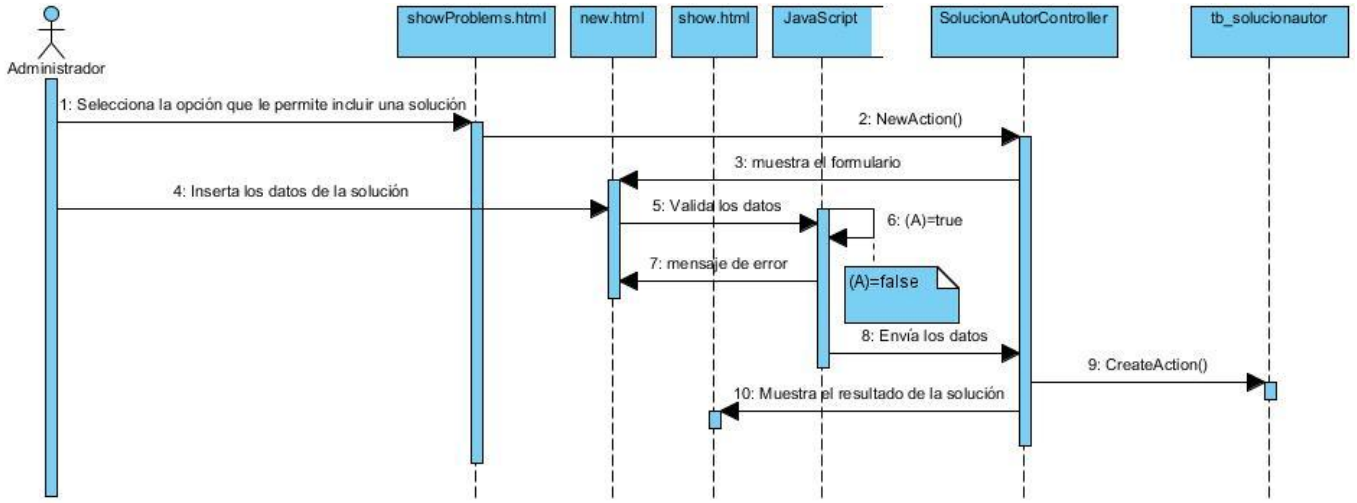


Figura 12 DS-Incluir Solución de Autor.

3.4 Modelo de despliegue

El modelo de despliegue define la arquitectura físca del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse elementos de software. Se utiliza como entrada principal en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño. (Jacobson, 2000)

3.4.1 Diagrama de despliegue (DD)

El diagrama de despliegue es utilizado para mostrar los nodos y conexiones del modelo de despliegue así como la asignación de los objetos a los nodos (Jacobson, 2000).

El diagrama que se presenta a continuación representa la distribución física del sistema AIA-SQL a través de nodos, está compuesto por una PC cliente que deberá tener instalado un navegador web, donde la comunicación entre ella y los servidores se llevará a cabo a través del Protocolo de Transferencia de Hipertexto (HTTP por sus siglas en inglés). El sistema contará además con un servidor web y un servidor de base de datos, estos servidores se comunicarán vía TCP-IP (familia de protocolos de Internet compuesta fundamentalmente por el Protocolo de Control de Transmisión, TCP por sus siglas en inglés, y el Protocolo de Internet, IP por sus siglas en inglés).

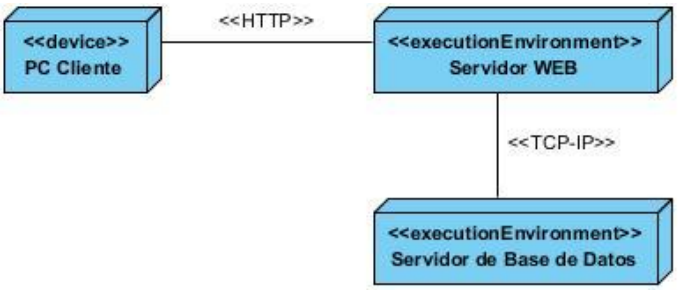


Figura 13 Diagrama de Despliegue (AIA-SQL)

3.5 Diseño de la Base de Datos

El diagrama entidad-relación es uno de los modelos más usados para diseñar bases de datos, este modelo se encuentra basado en dos conceptos fundamentales: entidades, que representan objetos sobre los cuales se desea guardar información y las relaciones, que constituyen las relaciones entre las entidades. (Ruiz, 2000)

A continuación se presenta el modelo de datos que contiene las entidades que serán utilizadas por las funcionalidades a desarrollar y las relaciones entre ellas, las mismas representan las tablas en la base de datos.

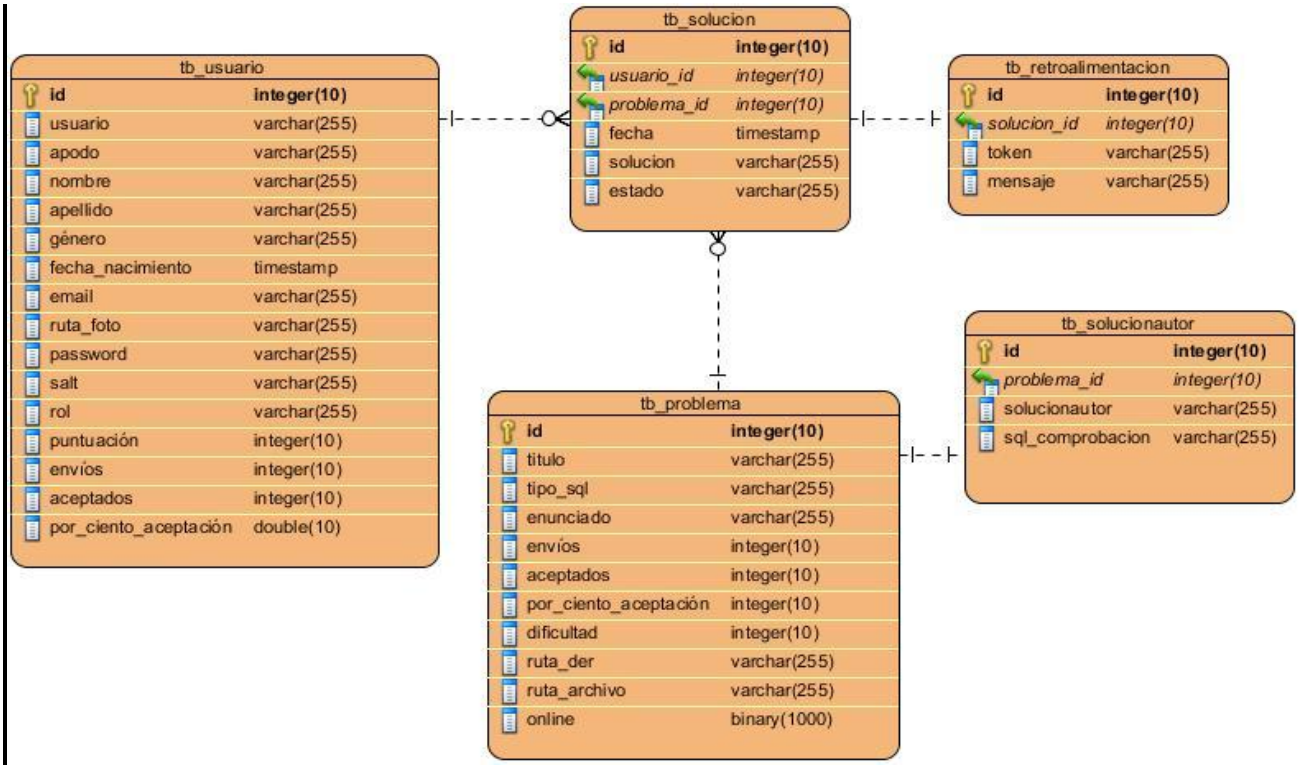


Figura 14 Diagrama Entidad-Relación (AIA-SQL)

3.5.1 Descripción de las tablas de la Base de Datos

A continuación se muestra la descripción de la tabla tb_problema creada para almacenar los datos de los problemas en el sistema. La descripción de las restantes tablas se muestra en el Anexo #7.

tb_problema		
Descripción: Esta tabla es la encargada de almacenar los datos de los problemas.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Campo que contiene el identificador del problema.
título	Varchar(255)	Campo que contiene el título del problema
tipo_SQL	Varchar(255)	Campo que contiene el tipo de problema.
enunciado	Text	Campo que contiene el enunciado del problema.
envíos	Integer(10)	Campo que contiene la cantidad de envíos realizados a un problema.
aceptados	Integer(10)	Campo que contiene la cantidad de envíos aceptados.
por_ciento_aceptación	Double precision	Campo que contiene el por ciento de aceptación de los envíos realizados.
dificultad	Integer(10)	Campo que contiene el valor que se le asigna a un problema teniendo en cuenta la dificultad.
ruta_der	Varchar(255)	Campo que contiene la ruta del diagrama entidad-relación.
ruta_archivosql	Varchar(255)	Campo que contiene la ruta del archivo SQL a utilizar en la solución.
online	Boolean	Campo que identifica si un problema está en línea o no.

Tabla 3 Descripción de la tabla tb_problema

3.6 Conclusiones Parciales

El desarrollo del flujo de análisis y diseño correspondiente a la metodología seleccionada, identificó los elementos que componen tanto la arquitectura como el diseño de la herramienta a implementar y la generación de los artefactos correspondientes para la posterior implementación. Como resultado se obtuvieron los diagramas generados tras el análisis y el diseño, se identificó además debido a las facilidades que brinda la arquitectura MVC como propuesta para la solución, se realizó el diagrama de despliegue identificando en cada caso los nodos físicos y las asociaciones de comunicación existentes

entre ellos y se desarrolló el diseño de la base de datos que se encargará de guardar la información con la que trabaja el sistema.

CAPÍTULO IV

Capítulo 4: Implementación y prueba

Introducción

Los artefactos generados durante la etapa de análisis y diseño constituyen el paso inicial para el desarrollo de la implementación, cuarto flujo de trabajo propuesto por RUP. El objetivo principal de esta etapa es la generación de clases, componentes u objetos ejecutables que se integrarán a un sistema. Durante el presente capítulo se presenta una descripción detallada a través de los diagramas de componentes del proceso de implementación de las funcionalidades que forman parte de la herramienta a desarrollar. Además se aplicarán las pruebas con el propósito de verificar la calidad del producto.

4.1 Retroalimentación y evaluación de las soluciones en el sistema AIA-SQL

Durante el desarrollo del sistema se establecen como necesidades fundamentales la evaluación de las soluciones enviadas por los estudiantes y la generación de retroalimentación asociada a los errores cometidos por los mismos. A continuación se describe brevemente el proceso de ejecución de ambos aspectos:

Al estudiante enviar una solución a uno de los problemas existentes en el sistema se crea una solución en estado ***Pendiente***. Como parte de las tareas a desarrollar por el sistema se encuentra la actividad Evaluar Semántica que analiza todas las soluciones en estado pendiente a partir de un conjunto de restricciones del lenguaje SQL creadas para este fin. Una vez analizada la solución, si la misma viola al menos una de las restricciones se califica como *Error de Compilación* generándose la retroalimentación perteneciente a la restricción que desencadenó el error. En caso contrario la solución pasa al estado ***Semántica***.

La otra tarea consiste en evaluar el código tomando como entrada las soluciones en estado Semántica. Para cada una de estas soluciones se corre el script SQL perteneciente al problema a solucionar, a su vez se ejecuta la solución del estudiante, de generar un error el sistema califica la solución de *Error de Compilación* y brinda una retroalimentación asociada a la excepción lanzada por PostgreSQL. De no lanzar error se guarda el resultado obtenido, se ejecuta la solución del autor para el mismo problema y se comparan los resultados. De no ser iguales se califica la solución como *Respuesta Incorrecta*, en caso contrario se califica como *Aceptada*.

4.2 Modelo de Implementación

El proceso de implementación parte como resultado del análisis y diseño de la propuesta de solución planteada, teniendo como objetivo llevar la arquitectura y el sistema como un todo. Describe como se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y los lenguajes de programación utilizados, y como dependen los componentes unos de otros. (Jacobson, 2000).

4.3 Diagrama de componente (DCOM)

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Comprende entre sus principales objetivos mostrar la dependencia lógica entre los distintos componentes del software y representar las relaciones entre los elementos que forman el código del sistema implementado (Jacobson, 2000). A continuación se presenta el diagrama de componentes propuesto para el CU Incluir Solución de Autor. Para consultar los restantes diagramas de componentes remitirse al Anexo #8.

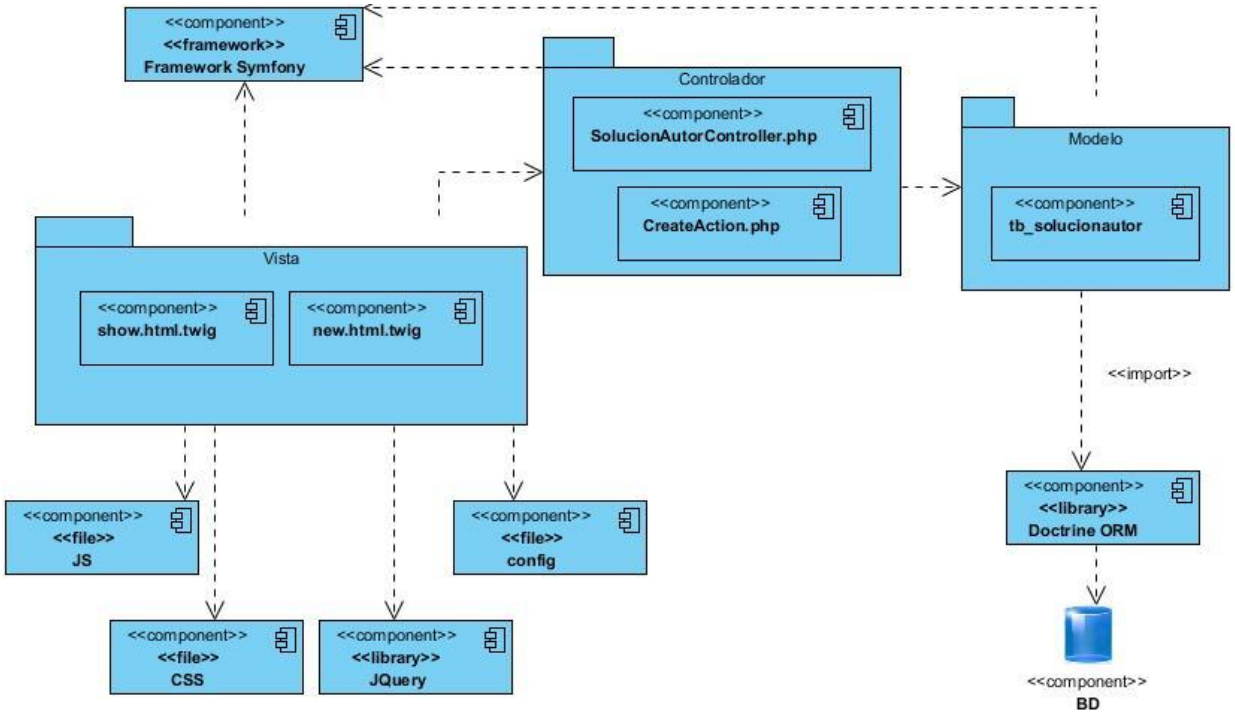


Figura 15 DCOM-Incluir Solución de Autor.

El flujo de trabajo destinado a las pruebas es el encargado de evaluar la calidad del producto que se está desarrollando y presenta como objetivos principales encontrar y documentar defectos en la calidad del software.

4.4 Modelo de Prueba

La ejecución de las pruebas al software es una de las etapas principales durante todo el ciclo de vida de la aplicación. Las mismas son una serie de actividades donde cada componente, o parte de la aplicación a comprobar, son ejecutados bajo determinadas condiciones. Su objetivo principal es evaluar y elevar la calidad. (Jacobson, 2000)

Las pruebas se aplican con diferentes objetivos y generalmente en distintos escenarios de trabajo. A continuación se describen las pruebas aplicadas al sistema AIA-SQL:

- **Pruebas de carga y estrés**: Las pruebas de carga permiten la simulación del acceso de muchos usuarios a un servidor al mismo tiempo, posibilitando observar el comportamiento de una aplicación bajo una cantidad de peticiones esperadas. Las pruebas de estrés se utilizan normalmente para romper la aplicación. Se va doblando el número de usuarios que se agregan a la aplicación y se ejecuta una prueba de carga hasta que se rompe. (Molyneaux, 2009)
- **Pruebas funcionales:** Las pruebas funcionales están centradas en comprobar que las funcionalidades descritas en el documento de requisitos del sistema se cumplen con la implementación realizada. A este tipo de pruebas también se les denomina pruebas de comportamiento o de caja negra, debido a que los analistas enfocan su atención a las respuestas del sistema de acuerdo a los datos de entrada y sus resultados en los datos de salida, los cuales se definen generalmente en los casos de prueba que se crean antes del inicio de las pruebas. (Palacio, 2009)
- **Pruebas de seguridad:** Las pruebas de seguridad permiten realizar una evaluación de los sistemas desde el punto de vista externo y sin conocimiento previo del mismo. 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. Verifican además que la aplicación y la infraestructura que la soporta no evidencian vulnerabilidades que puedan ser aprovechadas por terceros para un uso no deseado. (Salazar, 2013)

4.4.1 Métodos de Prueba

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas (Juristo, 2006). Estas técnicas se agrupan en:

- ✓ **Pruebas de caja blanca o estructural**: se basan en un minucioso análisis de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa (Juristo, 2006). Su uso posibilita la obtención de casos de prueba que garantizan, al menos una vez, que sean ejecutados todos los caminos independientes de cada módulo. Posibilita ejercitar

todas las decisiones lógicas en sus vertientes verdaderas y falsas. Permite, además, la ejecución de cada bucle con sus límites operacionales.

- ✓ **Pruebas de caja negra o funcional**: realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar. (Juristo, 2006)

Dentro de las pruebas de caja negra se incluyen las Técnicas de Prueba que serán descritas a continuación:

- ✓ **Partición de Equivalencia** divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.
- ✓ **Análisis de valores límites** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- ✓ **Grafos Causa-Efecto** permite validar complejos conjuntos de acciones y condiciones. (Pressman, 2005)

Con el objetivo de aplicar las pruebas de caja negra o funcional, es necesario apoyarse en el artefacto Casos de Prueba propuesto por la metodología de desarrollo seleccionada. Un caso de prueba especifica una forma de comprobar el correcto funcionamiento del sistema, en estos se incluyen las entradas, resultados y condiciones con la que se ha de verificar, constituyendo la guía principal para el probador. (Jacobson, 2000)

4.4.2 Diseño de Casos de Prueba (CP)

En los casos de prueba (CP) se incluyen la descripción de los principales escenarios, actores, posibles entradas, variables que intervienen en el proceso y flujo central donde se realiza el procedimiento. A continuación se presenta el CP propuesto para el CU Incluir Solución de Autor. Para consultar el resto de los CP remitirse al Anexo #9.

CP Incluir Solución de Autor.
Descripción General: El caso de uso inicia cuando un administrador crea un problema, para que este sea puesto *online* debe poseer una solución que será introducida por el administrador. Esta solución luego será usada

como guía en la evaluación de la solución del estudiante.
Condiciones de Ejecución:
El administrador debe haber creado un problema.
El sistema se encuentra en la interfaz Mostrar Problema.

SC1. Incluir Solución de Autor.

Escenario	Descripción	Respuesta del sistema	Flujo Básico
EC 1.1 Selecciona la opción **Solución Autor.**	Una vez en la ventana donde se muestran los datos del problema el administrador selecciona la opción **Solución Autor** que le permite dar solución al mismo.	Muestra una ventana que contiene un formulario para insertar la solución.	Portada/Gestionar Problema/ Ver Problema/ Solución Autor
EC 1.2 Inserta los datos y selecciona la opción **Enviar.**	Una vez insertada la solución en el formulario selecciona la opción **Enviar**.	Muestra información sobre el resultado de ejecutar la solución y actualiza el listado de problemas colocando el problema online.	Portada/Gestionar Problema/ Ver Problema/ Solución Autor/Enviar
EC 1.3 Selecciona la opción **Portada**	Si el usuario desea cancelar la operación en cualquier momento selecciona la opción **Portada** que se encuentra en la parte superior del menú.	Muestra la portada.	Portada/ Gestionar Problema/ Ver Problema/ Solución Autor/Portada
EC 1.4 Selecciona la opción **Limpiar**	Una vez introducidos los datos de la solución el estudiante puede borrarlos siempre y cuando no haya seleccionado la opción Enviar seleccionando *Limpiar.*	Elimina todos los valores asignados a los campos que ya han sido completados.	Portada/ Gestionar Problema/ Ver Problema/ Solución Autor/Limpiar
EC 1.5 El campo solución está vacío.	Al validar la solución se detecta que el campo solución está vacío.	Sombrea el campo vacío y muestra un mensaje de información. ‘’Rellene este campo’’	Portada/Problema/ Ver Problema/ Enviar/Enviar

Tabla 4 CP-CU Incluir Solución de Autor.

4.4.3 Resultados Obtenidos

4.4.3.1 Resultados de las pruebas funcionales

Con el objetivo de verificar el cumplimiento de los requisitos funcionales establecidos para la presente investigación se hace uso de las ***Pruebas de Caja Negra***, teniendo en cuenta la técnica de *partición por equivalencia*. Además, se hace uso de los casos de prueba generados durante este flujo de trabajo con el fin de detectar la mayor cantidad de no conformidades posibles en las funcionalidades del componente realizándose tres iteraciones de prueba. Para el seguimiento de todo el proceso de corrección de no conformidades se realiza una tabla, la misma contará con el Caso de Uso, la cantidad de no conformidades detectadas (**NC**), la cantidad de no conformidades significativas (**S**) y la cantidad de no significativas (**NS**) por cada Caso de Uso:

Casos de Uso	Iteración			Iteración 2			Iteración 3		
	NC	S	NS	NC	S	NS	NC	S	NS
CU-Autenticar Usuario	2	-	2	-	-	-	-	-	-
CU- Ver Perfil	1	-	1	-	-	-	1	-	1
CU- Configurar Perfil	1	1	-	-	-	-	-	-	-
CU- Registrar Usuario	2	1	1	-	-	-	-	-	-
CU- Ver mis Envíos	1	1	-	-	-	-	-	-	-
CU- Enviar Solución	2	1	1	1	1	-	-	-	-
CU- Ver problema	-	-	-	2	1	1	3	-	3
CU- Filtrar Problemas	-	-	-	1	1	-	-	-	-
CU- Buscar Usuarios	-	-	-	1	1		-	-	-
CU- Ver Estadísticas	1	-	1	1	-	1	-	-	-
CU- Ver Envíos	1	1	-	-	-	-	-	-	-
CU- Gestionar Usuario	2	1	1	-	-	-	-	-	-
CU- Gestionar Problemas	3	3	-	1	1	-	-	-	-
CU- Incluir Solución de Autor	-	-	-	-	-	-		-	-
CU- Editar Solución de Autor	1	1	-	-	-	-	-	-	-
CU- Ver Solución de Estudiantes	-	-	-	-	-	-	-	-	-
CU- Evaluar Solución	-	-	-	-	-	-	1	-	1
CU- Generar Retroalimentación	-	-	-	2	2	-	-	-	-

TOTAL	**17**	**10**	**7**	**9**	**7**	**2**	**5**	**0**	**5**

Tabla 5 Resultados de las pruebas de caja negra por iteraciones

A continuación se muestra un gráfico donde se puntualiza por iteraciones el total de no conformidades identificadas, el total de no conformidades resueltas y la cantidad de no conformidades pendientes. Para consultar las no conformidades detectadas por iteraciones remitirse al Anexo #10.

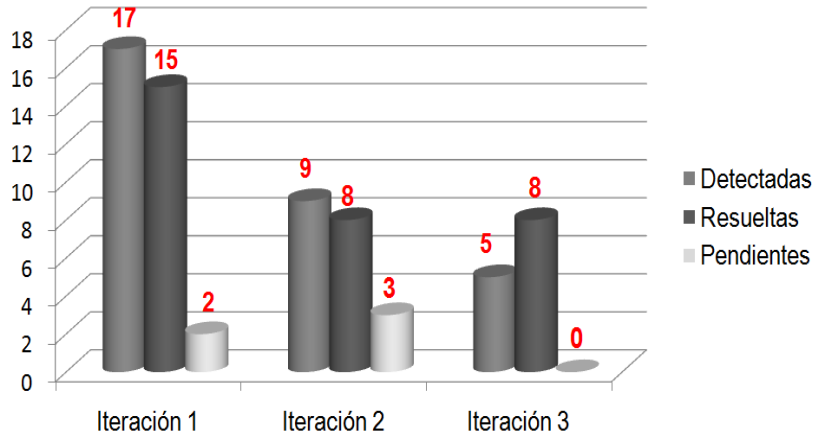


Figura 16 Resultados de las Pruebas de Caja Negra.

4.4.3.2 Resultados de las pruebas de seguridad

Con el propósito de comprobar entre otros aspectos, el acceso al sistema y a la información teniendo en cuenta para ello los roles que poseen los usuarios en el sistema se realizaron las ***pruebas de seguridad***. A continuación se muestra una tabla donde se especifica entrada o acción de usuario, resultado esperado del sistema y cumplimiento por parte del mismo, verificando de esta forma que el sistema cumple con estos aspectos (autenticación/autorización).

Prueba	Entrada o Acción de usuario	Resultado Esperado	Cumplimiento en el sistema
P 1	Usuario y Contraseña correcta.	El sistema permite el acceso al sistema, identificándolo correctamente y mostrándole la Portada de la sesión que le corresponde	Si

		teniendo en cuenta los roles.	
P 2	Usuario incorrecto y Contraseña correcta.	El sistema niega el acceso y muestra un mensaje informando el error.	Si
P 3	Usuario correcto y Contraseña incorrecta.	El sistema niega el acceso y muestra un mensaje informando el error.	Si
P 4	Usuario y Contraseña incorrectos.	El sistema niega el acceso y muestra un mensaje informando el error.	Si
P 5	Usuario y Contraseña correcta. Rol: Administrador.	Tiene acceso a la sesión de administración y tiene la posibilidad de interactuar con las funcionalidades definidas para el rol de administrador.	Si
P 6	Usuario y Contraseña correcta. Rol: Estudiante.	Tiene acceso a la sesión de los estudiantes y tiene la posibilidad de interactuar con las funcionalidades definidas para el rol de estudiante.	Si
P 7	Usuario se registra en el sistema.	Tiene acceso a la sesión de los estudiantes y tiene la posibilidad de interactuar con las funcionalidades definidas para dicho rol.	Si

Tabla 6 Pruebas de seguridad (autenticación y autorización)

Además se desarrollaron otras pruebas a la seguridad del sistema con la ayuda de la herramienta **Acunetix Web Vulnerability Scanner 8.0**, la misma permite analizar todas las posibles entradas de hackers que puedan producirse en una aplicación web, escaneando en busca de fallos y vulnerabilidades que impidan la seguridad de la web además de comprobar el estado del servidor en que se encuentre. La misma establece alertas de tipo: alta, media, baja e informacional. (ACUNETIX, 2013)

Se detectaron a través de la herramienta empleada un total de 137 no conformidades clasificadas en altas, medias, bajas e informacionales considerándose de mayor importancia para la investigación, las no conformidades catalogadas en altas y medias, las mismas fueron resueltas en su totalidad por el equipo de desarrollo.

A continuación se presenta un gráfico donde se puede apreciar la distribución de las mismas teniendo en cuenta el tipo de alerta:

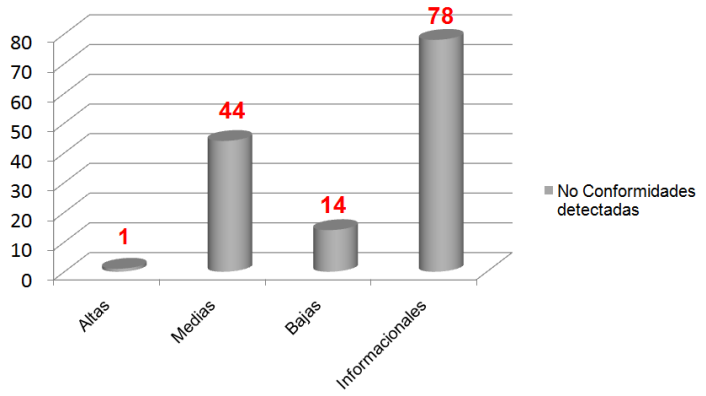


Figura 17 Resultados de las pruebas de seguridad.

4.4.3.3 Resultados de las pruebas de carga y estrés

Las pruebas de **carga y estrés** se desarrollaron con la ayuda de la herramienta **Apache JMeter**, la cual permite analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones web (Halili, 2008). Para el desarrollo de las mismas se hace uso de un ordenador con las siguientes características:

- Sistema Operativo Windows 7.
- Microprocesador Intel Celeron.
- Memoria RAM 2GB.

Los resultados obtenidos en las pruebas de carga se consideran satisfactorios, debido a que, los tiempos de respuesta del servidor se encuentran en un rango de tiempo de 1 a 3 segundos. A continuación se muestra un gráfico donde se representa el rendimiento obtenido para un total de 20, 50, 70 y 90 usuarios concurrentes. Para observar el informe agregado reportado por la herramienta remitirse al Anexo # 11.

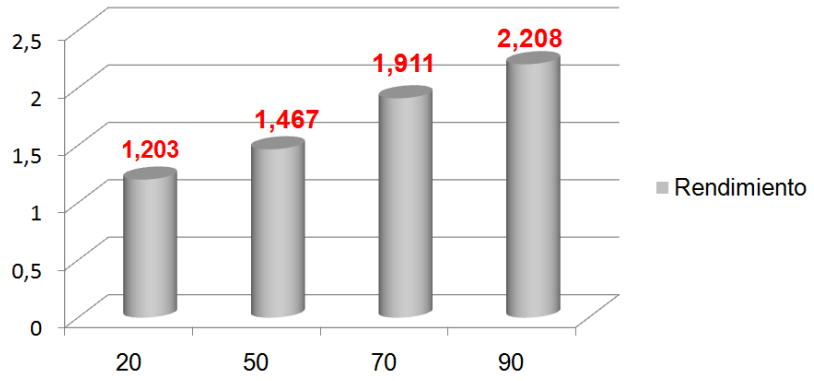


Figura 18 Resultado de las pruebas de carga.

Los resultados de las pruebas de estrés se consideran satisfactorios, debido a que, después de sobrepasar la cantidad de 100 usuarios con un total de 200, 250, 300 y 400 usuarios concurrentes, la propuesta se mantuvo estable prestando servicios sin incurrir en fallos, arrojando los resultados que se muestran a continuación:

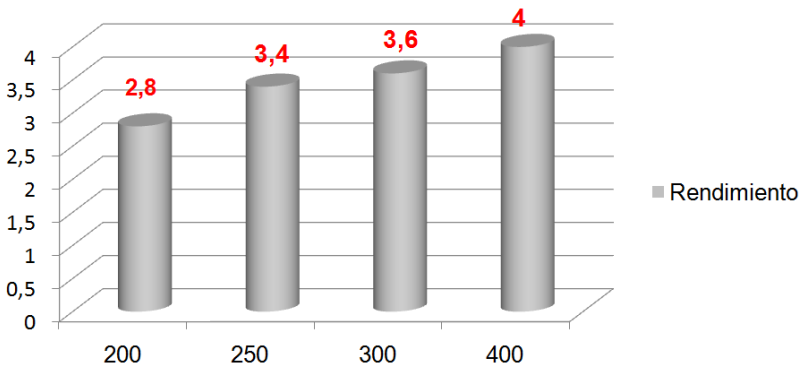


Figura 19 Resultado de las pruebas de estrés

Para observar el informe agregado reportado por la herramienta remitirse al Anexo # 11.

4.5 Conclusiones

Como resultado del desarrollo de este capítulo, se generaron los diagramas de componentes por cada caso de uso, así mismo se realizó la descripción de los casos de prueba con el objetivo de que los mismos sean usados para verificar el correcto funcionamiento de la aplicación. De modo general, se puede afirmar que una vez aplicadas las pruebas los resultados fueron satisfactorios ya que se identificaron a tiempo los

problemas presentados por la aplicación y se les suministró solución de forma inmediata. Finalmente se obtuvo un ambiente inteligente para el aprendizaje del lenguaje estructurado de consultas (AIA-SQL), que brinda el soporte a los estudiantes en la asignatura de Sistemas de Bases de Datos, mediante la entrega de retroalimentación.

CONCLUSIONES

Con la realización de la presente investigación se brinda solución a los objetivos trazados obteniéndose como principales resultados:

- ✓ La identificación de los aspectos teóricos como SQL, Jueces en Línea y los Sistemas Tutores Inteligentes fundamentó la necesidad de desarrollar un ambiente inteligente para el aprendizaje del lenguaje estructurado de consultas que garantice brindar soporte a los estudiantes.
- ✓ El análisis de las herramientas similares permitió identificar que los jueces en línea permiten la evaluación, pero no brindan la posibilidad de entregar retroalimentación al estudiante de sus actividades constituyendo esto una deficiencia.
- ✓ El desarrollo del flujo de análisis y diseño correspondiente a la metodología seleccionada, identificó los elementos que componen la arquitectura de la herramienta a implementar y la generación de los artefactos correspondientes para la posterior implementación.
- ✓ Mediante la puesta en práctica de las pruebas de caja negra, se pudo detectar y corregir los errores encontrados, logrando con esto verificar el perfecto funcionamiento de la solución propuesta.
- ✓ Se obtuvo un ambiente inteligente para el aprendizaje del lenguaje estructurado de consultas (AIA-SQL), que brinda soporte a los estudiantes en la asignatura de Sistemas de Bases de Datos, mediante la entrega de retroalimentación.

RECOMENDACIONES

A partir de los resultados obtenidos con la investigación los autores proponen las siguientes recomendaciones para futuros trabajos tomando como referencia el actual:

- Trabajar en una forma de representación de las restricciones de forma que puedan ser introducidas en el sistema sin hacer modificaciones al código.

- Realizar otras versiones tomando para la representación los otros formalismos existentes (Reglas de Producción y Sistemas Expertos) y comparar los resultados en función de varios indicadores.

- Estudiar el resultado que tiene sobre el aprendizaje, el funcionamiento del sistema en el marco de varios modelos pedagógicos.

BIBLIOGRAFÍA

1. ACUNETIX. 2013. ACUNETIX. Audit Your Website Security with Acunetix Web Vulnerability Scanner Disponible en:. [En línea] 2013. [Citado el: 23 de mayo de 2015.] http://www.acunetix.com/vulnerability-scanner/..
2. Aleven, V. 2010. Rule-Based Cognitive Modeling for Intelligent Tutoring Systems. [aut. libro] J. Bourdeau yR. Mizoguchi. (editores). R. Nkambou. *Advances in Intelligent Tutoring Systems. p. 33-62.* Springer-Verlag : s.n., 2010.
3. *Arias.* Francisco J.S. Ing., Jovani A Jiménez B., PhD., Demetrio A. Ovalle C. PhD. Pedagógico., Revista Avances en Sistemas e Informática.Una Aproximación Metodológica para la Construcción de Sistemas Tutoriales Adaptativos MultiAgente con Énfasis en el Modelo. 2007. 3, Medellín : s.n., 2007, Vol. 4. ISSN 16577663.
4. Bratko, Ivan. 2014. *Data-Driven Program Synthesis for Hint Generation in Programming Tutors.Proceedings of the International Conference on Intelligent Tutoring Systems. Springer.* 2014.
5. Brawner, K. W. y HOLDEN, H. K., et al. 2011. Understanding the Impact of Intelligent Tutoring Agents on RealTime Training Simulations. [aut. libro] K. W. BRAWNER y H. K., et al HOLDEN. *Proceedings of the Interservice/Industry Training, Simulation, and Education.* 2011.
6. Capps, H. L Burns y C. G. 1988. *Foundations of intelligent tutoring systems. Lawrence Erlbaum Associates, Inc.* 1988.
7. Carmona, Juan García. 2012. *Solid y GRASP.Buenas prácticas hacia el éxito en el desarrollo de software.* 2012.
8. Casany, Carme Martín Alberto Abelló Xavier Burgués Carme Quer M. José. 2013. *LEARN-SQL: Una herramienta para el soporte al aprendizaje semipresencial en el ámbito de las bases de datos LEARN-SQL: A blended learning tool for the database area.* 2013.
9. Castillo, Enrique José Altuna, Lisandra Guibert Estrada , Vivian Estrada Sentí. 2014. 4, Método para la construcción del modelo de dominio en un Tutor. Habana : Revista Cubana de Ciencias Informáticas, 2014, Vol. 8. ISSN: 2227-1899 | RNPS: 2301.
10. Coca, Guadalupe Hernández. 2011. *Lenguaje Estructurado de Consulta SQL.* Hidalgo : Universidad autonoma del estado de Hidalgo, Escuela superior Huejutla Area Académica: Sistemas Computacionales, 2011.
11. Community. 2013. An Introduction to NetBeans. Welcome to the NetBeans. [En línea] 2013. [Citado el: 10 de enero de 2015.] https://netbeans.org/about/index.html..
12. Darie, Christian. 2003. *"The Programmer's Guide to SQL".* United States of America: Apress : s.n., 2003. ISBN (pbk): 1-59059-218-2.).
13. Dekeyser, Raadt M Lee T.Y S. 2007. *"Computer Assisted Assessment of SQL query Skills".* 2007.
14. Eguiluz, Javier. 2011. *Desarrollo web ágil con Symfony 2.* 2011.

15. Euguiluz, Javier. 2005. *Introducción a Ajax.* 2005.

16. Fabien Potencier, François Zaninotto. 2008. symfony 2 guia definitiva. [En línea] 2008. [Citado el: 15 de octubre de 2014.] http://librosweb.es/libro/symfony_1_2/capitulo_2/el_patron_mvc.html. ISBN: 0-32112-742-0.

17. Fernandez, Amarán. 2010. *Sistema de Gestión de Datos Geológicos: módulo. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.* La Habana, Cuba : Consultada 10 diciembre 2014., 2010.

18. Ferreira, José Osvaldo Chaves Angélica Castro Marcos Vinicius Lima Karl Hansimuller Alelaf. 2013. *Uma Ferramenta Baseada em Juízes Online para Apoio Atividades de Programação de Computadores no.* CINTED-UFRGS : V. 11 Nº 3, 2013.

19. GAMMA, E., HELM, R., JOHNSON, R. y VLISSIDES, J. 2005. *Design Patterns - Elements of Reusable Object Oriented Softwar.* 2005. ISBN 0-201-63361-2..

20. Garis, Alberto Cortez y Ana. 2012. *Aplicación de Perfiles UML en la Especificación de Patrones de Comportamiento.* Argentina : Argentine Symposiuum on Software Engineering, 2012. ISSN 1850-2792.

21. GAUCHAT, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Marcombo : s.n., 2012. ISBN 8426717829. .

22. Giraffa, L. M. M. 1997. *Seleçao e adoçao de estrategias de ensino em Sistemas Tutores Inteligentes .* Porto Alegre: CPGCC/UFRGS. : s.n., 1997.

23. Guzman, E. de los Riscos. 2005. *Un modelo de evaluación cognitiva basado en TAI para el diagnóstico en STI.* España : Universidad de Málaga., 2005.

24. Halili, Emily. 2008. *Apache JMeter.* 2008. ISBN:1847192955 9781847192950 .

25. Hans-Erik Eriksson, Magnus Penker. 2000. *Business Modeling with UML:Business Patterns at Work.* Canadá : John Wiley & Sons, Inc, 2000. ISBN: 0471295515.

26. Hillsdale, N.J. 2009. *"SISTEMA TUTOR PARA ENSEÑANZA DEL LENGUAJE SQL".* Bolivia : Lawrence Erlbaum Associates, 2009.

27. Holdener, Anthony T. 2008. *AJAX The definitive Guide.* Gravenstein Highway North : O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2008. ISBN: 978-0-596-52838-6.

28. Ibarra, Eduardo Samaniego Jennifer Rodríguez Álava Cinthia Vera. 2013. *TALLER DE APLICACIONES WEB. XAMPP.* s.l. : UNIVERSIDAD TÈCNICA ESTATAL DE QUEVEDO, 2013.

29. Jacobson, Grady BOOCH James RUMBAUGH Ivar. 2000. *El Lenguaje unificado de modelado. Manual de referencia.* Madrid : Pearson Educación : Disponible en: http://bibliodoc.uci.cu/pdf/reg03050.pdf005.131-Rum-L., 2000. ISBN 84-7829-037-0.

30. Jacobson, Grady Booch James Rumbaugh Ivar. 2000. *El proceso unificado de desarrollo de software.* Madrid : Pearson Eduaction S.A., 2000. ISBN 84-7829-036-2.

31. Juristo, Natalia, Ana M. Moreno, Sira Vega. 2006. Técnicas de evaluación de software. *GrISE (Grupo de investigación de Ingeniería de Software Empírica).* [En línea] 17 de octubre de 2006. [Citado el: 23 de febrero de 2015.] http://www.grise.upm.es/sites/trs/1/pdf/Documentacion_Evaluacion_7.pdf.

32. Kabir, Mohammed J. 2003. *La biblia del servidor Apache.* s.l. : Anaya Multimedia, 2003, 2003. ISBN 8441514682, 9788441514683.

33. Kurnia, Andy, Lim, Andrew y Cheang, Brend. 2001. *Online Judge.* s.l. : Elsevier Science Ltd. Oxford Computers & Education archive, 2001. doi>10.1016/S0360-1315(01)00018-5.

34. LANCKER, Luc Van. 2009. *XHTML y CSS - Los nuevos estandares del codigo fuente.* s.l. : Ediciones ENI, 2009. ISBN 9782746047426. .

35. Larman, Craig. 2003. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. .* s.l. : Pearson Educación, 2003. ISBN 9788420534381. .

36. Martinez, Rafael. 2014. Sobre PostgreSQL. . [En línea] 2014. [Citado el: 30 de noviembre de 2014.] www.postgresql.org.es/sobre_postgresql.

37. Masó, Josep Soler. 2010. *ENTORNO VIRTUAL PARA EL APRENDIZAJE Y LA EVALUACIÓN AUTOMÁTICA EN BASES DE DATOS.* Universidad de Girona : Dipòsit legal: GI-1453-2010, 2010. ISBN: 978-84-694-0260-3.

38. Mehd ACHOUR. 2006. *PHP Documentation Group. PHP Manual.* [En línea] 2006. [Citado el: 20 de octubre de 2014.] Disponible en: http://www.php.net/docs.php.

39. Mestras, Juan Pavón. 2013. *Bootstrap 3.0, Aplicaciones Web/Sistemas Web.* Dep. Ingeniería del Software Facultad de Informática Universidad Complutense Madrid : UCM 2013-14, 2013.

40. Miranda, Osmani A. 2004. *El Lenguaje SQL, su aplicación en el acoplamiento de tablas en Base de Datos Revista Electrónica Granma Ciencia.* La Habana : Escalona INSTITUCIÓN: Jefatura Provincial del MININT. Carretera Central, Vía Santiago de Cuba km 1 ½ Rpto. Viviendas Campesinas. Bayamo. Granma. E-MAIL, 2004. ISSN 1027-975X.

41. Mitrovic, A. 1998. *“Learning SQL with a computerized Tutor”, Proc. of SIGCSE’98, pp.307–311,.* 1998.

42. Mitrovic, A., B. Martin, et al. 2009. *"ASPIRE: An authoring system and deployment environment for constraint-based tutors". International Journal of Artificial Intelligence in Education.* 2009.

43. Mitrovic, Anija. 2003. *An intelligent SQL tutor on the web.* 2003.

44. Mitrovic, Antonija. 2003. *All rights reserved An Intelligent SQL Tutor on the Web Antonija Mitrovic, Intelligent Computer Tutoring Group.* s.l. : International Journal of Artificial Intelligence in Education 13, 2003. ISBN 1560-4292/03/.

45. Mitrovic, Antonija. 2012. *"Fifteen years of constraint-based tutors: what we have achieved and where we are going". User Modeling and User-Adapted Interaction.* s.l. : DOI 10.1007/s11257-011-9105-9, 2012.

46. Molyneaux, Ian. 2009. *The art of Application Performance Testing.* 2009. ISBN: 978-0-596-52066-3 ISBN 10: 0-596-52066-2.

47. Moritz, S. y G. Blank. 2008. *Generating and Evaluating Object-Oriented Designs for Instructors and Novice Students. Proceedings of the Intelligent Tutoring Systems for IllDefined Domains: Assessment and Feedback in Ill-Defined Domains. Montreal.* 2008.

48. Ohlsson, S. y Mitrovic, A. 2007. *Fidelity and efficiency of knowledge representations for intelligent tutoring.* 2007.

49. Ojeda, Isaac Quintana, Mariana Martinez Almeida. 2013. *Manual de Jquery.* 2013.

50. Orlowska, M.E. Lin S.W Sadiq. 2004. *SQLator: An online SQL learning workbench.* New York, U.S.A : ACM Press editor: R. Boyle, 2004. ISBN 1-58113-836-9.

51. Pacheco, Doctrine Project Team :: Traducido por Nacho. 2011. *Doctrine 2 ORM Documentation Release 2.1.* 2011.

52. *Palacio.* software, Método para generar casos de prueba funcional en el desarrollo de. 2009. 15, Liliana González Palacio Medellín : Rev. ing. univ. Medellín, 2009, Vol. 8. ISSN 1692-3324.

53. Pérez, Javier EGUÍLUZ. 2009. *Introducción a JavaScript.* 2009.

54. Perrenoud, Ph. 2004. *Diez Nuevas Competencias para Enseñar.* Barcelona : Barcelona : Graó, 2004.

55. Polanco, Ignacio. 2007. *LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN (TIC) EN LA EDUCACIÓN: RETOS Y POSIBILIDADES.* s.l. : Fundación Santillana, 2007.

56. Polson, M. C., & Richardson, J. J. 1988. *Foundations of Intelligent Tutoring Systems.* s.l. : Hillsdale, NJ: Lawrence Erlbaum Associates Publishers, 1988.

57. Pressman, Roger S. 2005. *Ingeniería del Software: Un Enfoque Práctico.* 2005. ISBN: 9701054733.

58. Puente, Ledesma Rodríguez Yenisleydis Boza Roget Yunier Robert Lobo Armando García De La. 2011. *Extensión de la herramienta Visual Paradigm for UML para el soporte al Desarrollo Dirigido por Modelos con Ext JS.* La Habana : s.n., 2011.

59. RAE, Real Academia Española. 2012. [En línea] 2012. [Citado el: 13 de noviembre de 2014.] http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&BUS=3&LEMA=metodolog%C http://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&BUS=3&LEMA=metodolog%Chttp://buscon.rae.es/draeI/SrvltGUIBusUsual?TIPO_HTML=2&BUS=3&LEMA=metodolog%Chttp://buscon.rae.es.

60. Ribeiro, Joanna Santos Admilson R. L. 2011. *JOnline: proposta preliminar de um juiz ensino de programação.* Avenida Marechal Rondom, S/N, Jardim Rosa Elze : UFS - Universidade Federal de Sergipe CEP 49.100-000 São Cristóvão, SE, 2011. ISSN: 2176-4301.

61. Rodés, Juan Daniel Santana. 2010. *JUEZ EN LÍNEA DE BASE DE DATOS DATA BASE ONLINE JUDGE.* Habana : s.n., 2010.

62. Rodés, Santana Rodés Juan Danie Yaniel Blanco Fernández Ráiner Cárdenas Alvarez Juan Daniel Santana. 2013. *JUEZ EN LÍNEA DE BASE DE DATOS DATA BASE ONLINE JUDGE.* s.l. : Universidad de las Ciencias Informáticas,, 2013.

63. Rodríguez, Erick. 2012. Patrones del “Gang of Four” [online]. [En línea] 2012. [Citado el: 17 de marzo de 2015.] http://is.ls.fi.upm.es/docencia/proyecto/docs/patrones_gof.pdf.

64. Ruiz, Francisco. 2000. *Modelo Entidad/Realación.* s.l. : UCLM-ESI, 2000.

65. Sadiq S, Orlowska M., Sadiq W., Lin J. 2004. *"SQLator—an online SQL learning workbench.* 2004.

66. Salazar, Edgar D. 2013. *Pruebas de Seguridad en aplicaciones web segun OWASP.* Venezuela : s.n., 2013.

67. Sleeman, J. S. Brown D. 1982. *Intelligent tutoring systems.* s.l. : Academic Press, Inc., 1982.

68. Sommerville, Iam. 2005. *Ingeniería de Software Séptima Edición.* Madrid : Pearson Educación, 2005. ISBN 84-7829-074-5.

69. SWEAT, Jason E. 2005. Guide to PHP Design Patterns [online. *php|architect's]. Canada.* [En línea] 2005. [Citado el: 14 de marzo de 2015.] ISBN 0-9735898-2-5.

70. Thornton, Mark Otto Jacob. 2014. *Bootstrap 3, el manual oficial.* 2014.

71. Vanlehn, K., C. Lynch, et al. 2005. *"The Andes Physics Tutoring System: Lessons.* 2005.

72. Vázquez, MSc. Tomás Orlando Junco. 2012. *UN JUEZ EN LÍNEA AJUSTADO A LAS NECESIDADES DE LA DOCENCIA.* La Habana : Noviembre, 2012.

73. Wenger, E. 1987. *Artificial intelligence and tutoring systems. Computational and Cognitive Approaches to the Communication of Knowledge.* s.l. : Los Altos C. A. Morgan and Kaufman., 1987.

74. Woolf, B. 1984. *Context Dependent Planning in a Machine Tutor. Ph.D. Dissertation, University of Massachusetts,.* Amherst, Massachusetts : s.n., 1984.

75. Woolf, B. P.. 2009. *Building Intelligent Interactive Tutors.* 2009.

76. Zulma, Fernando J. Lage y. 2010. *Modelo de Sistemas Tutor Inteligente distribuido para educación a distancia.* Buenos Aires : Cataldi LIEMA - Laboratorio de Informática Educativa y Medios Audiovisuales., 2010.