Universidad de las Ciencias Informáticas

Facultad 4

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

Componente para la gestión de restricciones y seguimiento de los estudiantes en el Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL)

Autor:

Alexander Cutiño Rivera

Tutores:

Ing. Arcel Labrada Batista.

Ing. Yaritza Bárbara González Ramírez.

Ciudad de la Habana

"Tienes que confiar en algo – en tu instinto, el destino, la vida, el karma, lo que sea. Esta perspectiva nunca me ha decepcionado y ha hecho toda la diferencia en mi vida."

Steve Jobs

Declaración de Autoría

Declaro ser el único autor del presente trabajo de diploma y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Autor

_____

Alexander Cutiño Rivera

_____ _____

Ing. Arcel Labrada Batista

Ing. Yaritza Bárbara González Ramírez

Dedicatoria

A mis padres Minerva, Jorge, Yami y Ramón por apoyarme durante todos estos años, por ser mi ejemplo, por ser mis guías, mi logro es de ustedes.

A mi abuela por ser esa persona especial que siempre estuvo ahí para mí, en las buenas y en las malas, que me cuido como si fuera su hijo, por ser la razón de mi esfuerzo, esta tesis es para ti.

A mi hermana que ha sido un ejemplo para mí aun siendo la pequeña, por todos los buenos momentos que hemos pasado juntos y por estar ahí para mí cuando lo he necesitado.

A mis tías y tíos Sofía, Clari, Laura, Damelis, Benancio, Carmencita y Osvaldo por ayudarme en todo lo que han podido y brindarme su apoyo, gracias.

A Lisbeth y Nela por ser las hermanas mayores que nunca tuve y apoyarme cuando hiciera falta.

A Adonis y Mabel por toda la ayuda que me han dado, por los buenos momentos que hemos compartido, por ser ejemplos a seguir, por ser mi hermano mayor y a ambos por demostrarme que siempre he podido contar con ustedes.

A mis primitos, Ivett, Andy, Alain y el Cotunto por las alegrías que me han dado y todos los momentos que hemos compartido.

A mi abuelo Mimo y Papá Pedro que aunque no estén ya con nosotros sé que estarían orgullosos de mí, este logro es para ustedes también.

Agradecimientos

Durante todos estos años de estudio muchas personas han llegado a mi vida sin las cuales no me hubiera sido posible cumplir mi meta, sin ustedes no hubiera sido posible este momento, por eso quiero a agradecerle a todos por ese aporte que han hecho en mi vida y mi carrera.

A mi familia que ha sido el apoyo y ejemplo que me han permitido seguir adelante, que me ha apoyado en los momentos buenos y malos, gracias.

A todos mis amigos, sin ustedes estos años de universidad no habrían sido tan geniales, han sido, son y serán mi familia, gracias a todos.

A mi tutor Arcel por toda la ayuda que me ha brindado que me ha permitido lograr esta meta.

A mi tutora Yaritza que más que tutora compañera de tesis por ayudarme más de lo que debías, muchísimas gracias sin ti no lo habría logrado.

A todos aquellos que de una forma u otra me ayudaron a alcanzar mi meta.

Resumen

En Cuba el empleo de la tecnología abre grandes posibilidades de apoyo a los procesos de aprendizaje debido a que hace posible la implementación de aplicaciones o herramientas que surgen en apoyo al Proceso de Enseñanza Aprendizaje (PEA). En la Universidad de las Ciencias Informáticas como parte de los trabajos de diploma de los estudiantes que cursan el último año de la carrera se desarrollan varios sistemas que contribuyen en su mayoría al PEA. El Ambiente Inteligente para Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL) es uno de los sistemas desarrollados con este fin. El mismo tiene como propósito brindar ayuda a los estudiantes durante el desarrollo de las actividades en la asignatura Sistemas de Bases de Datos. El objetivo de la presente investigación es incorporar a dicho sistema un componente que permita la gestión dinámica de las restricciones y la generación 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 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, restricciones, retroalimentación, Jueces en Línea, Sistemas Tutores Inteligentes.

Índice de contenido

Introducción

El desarrollo paulatino de la sociedad, ha demandado cambios en el sector educacional para lograr mayor interactividad y acceso a la información (Entonado, 2001). Debido a que la inclusión de nuevos contenidos académicos en el ámbito universitario es innegable, la vinculación de las tecnologías en el proceso formativo ha dado paso al surgimiento del aprendizaje mediante el uso de la tecnología (Bustamante, 2013) y como resultado se han introducido las Tecnologías de la Información y las Comunicaciones (TIC) en el Proceso Enseñanza-Aprendizaje (PEA) para apoyar los esquemas tradicionales de la enseñanza presencial o a distancia.

La incorporación de la tecnología en la educación provee de herramientas que mejoran el entendimiento y asimilación de los contenidos de las asignaturas, brindándole al profesor un medio para desarrollar con mayor calidad su papel en el PEA. Hoy en día, es común que se aprovechen programas de ordenador para un aprendizaje más dinámico y autoguiado, debido a que dentro de este nuevo esquema de aprendizaje estos son muy utilizados tanto por los profesores como instrumentos de trabajo, medios didácticos o canales de comunicación (Entonado, 2001).

En Cuba el empleo de la tecnología abre grandes posibilidades de apoyo a los procesos de aprendizaje debido a que hace posible la implementación de aplicaciones o herramientas que surgen en apoyo al PEA. Es importante destacar que gracias al uso de estas herramientas los estudiantes asimilan un mejor aprendizaje (Almaguer, 2011). Como parte de este proceso han sido realizadas un conjunto de acciones para contribuir a la actualización del Sistema Educativo y responde a las necesidades de la utilización de las nuevas tecnologías.

Algunas de las universidades del país han sido las encargadas de realizar el mayor aporte en dicho proceso mediante la vinculación estudio-trabajo, entre las cuales se pueden mencionar: la Universidad de la Habana (UH) con la Multimedia educativa para el perfeccionamiento del proceso enseñanza-aprendizaje de la asignatura Biología Celular, y la Universidad de las Ciencias Informáticas (UCI). Esta última, cuenta con centros de desarrollo e investigación asociados a las diferentes facultades, uno de ellos es el Centro de Tecnologías para la Formación (FORTES) ubicado en la Facultad 4, el cual tiene como objetivo producir softwares educativos.

En dicho centro se han desarrollado varias aplicaciones para apoyar el PEA, entre ellas se puede mencionar: personalizaciones realizadas a MOODLE para el apoyo a las diferentes asignaturas que se imparten en la UCI así como para otras esferas de la sociedad. También la Plataforma educativa ZERA y la herramienta de autor CRODA. Además, han sido desarrollados como parte de los trabajos

de diploma de los estudiantes que cursan el último año de la carrera varios sistemas que contribuyen en su mayoría al PEA. El Ambiente Inteligente para Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL) es uno de los sistemas desarrollados con este fin. El mismo tiene como propósito brindar ayuda a los estudiantes durante el desarrollo de las actividades en la asignatura Sistemas de Bases de Datos.

El sistema mencionado anteriormente permite la evaluación automática de los ejercicios realizados por los estudiantes en la asignatura y a la vez brinda información sobre los errores cometidos durante su desarrollo. Para llevar a cabo la evaluación de las soluciones enviadas por los estudiantes este hace uso de un motor de evaluación el cual funciona como un Juez en Línea y es el encargado de realizar este proceso de manera transparente a los usuarios, mediante la utilización de técnicas de programación. Durante este proceso, en caso de existir algún error en la solución enviada por el estudiante, el sistema brinda una retroalimentación con el objetivo de que el mismo pueda alcanzar su propósito. Con el objetivo de que el sistema proponga una retroalimentación lo más cercana posible a las necesidades formativas del estudiante se utiliza un Sistema Tutor Inteligente basado en restricciones que utiliza técnicas de Inteligencia Artificial para representar el conocimiento e interactúa con los estudiantes para enseñárselo y generar dicha retroalimentación. El Sistema Tutor Inteligente funciona mediante un Analizador Semántico que es el encargado de analizar las respuestas de los estudiantes y generar una retroalimentación al encontrar un error en la solución.

Durante el trabajo diario con el sistema AIA-SQL se detectaron deficiencias que afectan su correcto funcionamiento, las mismas fueron identificadas y enumeradas como se muestra a continuación:

1. Las restricciones utilizadas por el sistema tutor inteligente para la generación de retroalimentación a las respuestas de los estudiantes se encuentran escritas de forma estática en el código fuente de la aplicación, lo que dificulta la modificación, visualización, eliminación e inserción de nuevas reglas a menos que se tenga acceso a este código y conocimientos de programación.
2. La cantidad de restricciones presentes en el sistema es de solo 7, a pesar de que existe constancia de la existencia de más de 100 en ambientes educativos, lo que afecta la completitud del sistema.
3. La evaluación de las restricciones es realizada por el analizador semántico el cual es el núcleo central del motor de evaluación, por lo que cualquier cambio realizado en las mismas puede afectar el funcionamiento de todo sistema.

4. No se almacena un historial de las retroalimentaciones otorgada por el motor de evaluación lo que evita la toma de decisiones en cuanto al valor formativo de la retroalimentación y la validez de las restricciones.

Teniendo en cuenta la situación planteada anteriormente se define como **Problema a resolver:** ¿Cómo lograr el trabajo dinámico con las restricciones del Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL) y un mejor seguimiento del resultado de los estudiantes?

Con el fin de dar solución al problema de la investigación se delimita como **Objeto de Estudio**: El funcionamiento de los Sistemas Tutores Inteligentes.

El **Campo de Acción** se enmarca en el: Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta.

Definiéndose como **Objetivo General**: Desarrollar un componente que permita el trabajo dinámico con las restricciones del Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL) y genere la retroalimentación.

El desarrollo del presente trabajo estará guiado por las siguientes tareas de **investigación, ingeniería** e **implementación y prueba**:

Investigación:

- Estudio de los sistemas similares a nivel nacional e internacional que permitan la gestión dinámica de restricciones.
- Estudio del uso de patrones para el diseño de aplicaciones.
- Estudio de los principales conceptos y tecnologías que se requieran para el desarrollo del componente para la gestión de restricciones y seguimiento de los estudiantes.
- Estudio de la metodología de software a utilizar para el desarrollo del componente para la gestión de restricciones y seguimiento de los estudiantes.

Ingeniería:

- Identificación y especificación los requerimientos con los que debe cumplir el componente para la gestión de restricciones y seguimiento de los estudiantes.
- Identificación y aplicación patrones de diseño en el desarrollo de la aplicación.
- Obtención los artefactos generados según el proceso de desarrollo aplicado para la realización de una propuesta de solución de un producto de software.
- Diseño de los casos de prueba a emplear para la comprobación de la aplicación.

Implementación y prueba:

- Desarrollo de un componente que permita la gestión de restricciones y seguimiento de los estudiantes en el Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL).
- Realización de pruebas funcionales al componente para comprobar su funcionamiento.

Para la realización de las tareas expuestas anteriormente, se hacen uso de diferentes métodos científicos, los cuales a su vez se dividen en teóricos y empíricos. A continuación se enuncian los métodos utilizados:

Los métodos teóricos utilizados para el desarrollo de la investigación son el **análisis histórico-lógico**, para el estudio de otras soluciones o aplicaciones similares con la investigación, así como las metodologías de desarrollo, lenguajes de programación y framework (marcos de trabajo) a utilizar en el desarrollo del componente, el **analítico-sintético** para identificar, analizar y seleccionar los conceptos y definiciones más importantes relacionadas con el tema de la investigación en aras de crear el marco teórico del trabajo que permita generar una propuesta de solución adecuada a la situación planteada, **análisis documental** para realizar consultas a la literatura especializada en el tema abordado en la presente investigación y el **hipotético-deductivo** en la propuesta de nuevas líneas de trabajo a partir de los resultados obtenidos.

El método empírico utilizado fue la **observación,** con el objetivo de observar funcionamiento de los Sistema de gestión, Jurados Online y Sistema Tutor Inteligente, así como sus principales problemas.

El presente trabajo de diploma está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones generales, recomendaciones, referencias bibliográficas y anexos.

- **Capítulo 1:** Fundamentación Teórica. Recoge todo lo que concierne al objeto de estudio, se analizan los principales conceptos sobre Juez en Línea y Sistemas Tutores Inteligentes a través de una revisión bibliográfica, se observaron sus ventajas, limitaciones y los requisitos necesarios a tener en cuenta en su elaboración. Se realiza un estudio de las metodologías y herramientas de desarrollo que se pueden utilizar para dar solución al problema planteado, así como la fundamentación de su uso.
- **Capítulo 2:** Propuesta de solución. Se exponen las características del sistema a implementar a partir de un levantamiento de las necesidades que debe cumplir la solución, definiéndose los requisitos funcionales y no funcionales de la misma. Se generan los artefactos correspondientes a la metodología de desarrollo a utilizar, que incluyen los diagramas pertinentes, lo que permite el modelado de la solución de manera visual.

- **Capítulo 3:** Análisis, diseño, implementación y prueba. Este capítulo abarca todo lo relacionado con el desarrollo del componente. Se reflejan aspectos relacionados con la construcción de la solución propuesta, se evidencian los patrones de diseño y los estándares de codificación. Además se definen las pruebas y se elaboran los casos de pruebas a realizarle al software, para así verificar la integridad del mismo y el cumplimiento de los requisitos definidos por el cliente. Finalmente se realizan las pruebas para la validación de la solución propuesta.

Capítulo I: Fundamentación Teórica

1 Introducción

En el presente capítulo se abordan los principales conceptos referentes a la investigación así como los principales aspectos investigados que constituyen el fundamento teórico para el desarrollo del componente en cuestión. 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 solución.

1.1 Conceptos asociados al dominio del problema

Para desarrollar la solución se hace necesario abordar brevemente el conocimiento de los conceptos referentes a Juez en Línea y Sistemas Tutores Inteligentes los cuales fueron analizados en el desarrollo del Ambiente Inteligente para el Aprendizaje Lenguaje Estructurado de Consulta (AIA-SQL) puesto que juegan un papel importante en el desarrollo componente, así como se hace imprescindible la explicación conceptual del término retroalimentación en los Sistemas Tutores Inteligentes basados en restricciones y una breve descripción del AIA-SQL.

1.1.1 Juez en Línea

Un Juez en Línea es una 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. Se caracterizan por tener una interfaz bien definida para la interacción con el usuario y un alto nivel de disponibilidad, lo que permite el libre acceso a la aplicación en todo momento a las personas registradas (Junco, 2012).

1.1.2 Sistemas Tutores Inteligentes

El desarrollo de Sistemas Tutores Inteligentes (STI) tiene como objetivo crear herramientas que enseñan un dominio de conocimiento en forma similar a como lo haría un tutor humano experimentado. Para lograr lo el sistema debe ser capaz de modelar el estado de conocimiento y recomendar los materiales e intervenciones más adecuadas para cada estudiante (Castillo, 2015).

Formalmente un Sistema Tutor Inteligente puede definirse como: *"Sistema que 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"* (Wegner, 1987).

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).

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.

1.1.3 Modelo de 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).

Existen tres formas fundamentales que se usan para representar y razonar sobre el conocimiento del dominio: modelos basados en reglas, basados en restricciones y basados en 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, Kaleb, Lynch, Charles, 2005).
- Los **modelos basados en restricciones** expresan las condiciones que deben cumplir todas las soluciones correctas mediante el uso de restricciones. El sistema diagnostica los errores por la violación de restricciones y entrega retroalimentación según la restricción que fue violada (Mitrovic, 2012).
- 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).

1.1.4 Retroalimentación en los STI basados en restricciones

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).

Una 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). Las restricciones se especifican de como se muestra a continuación:

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).

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 (Mitrovic, 2003).

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.

Imagen 1. Ejemplo de una restricción en un STI

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

Imagen 2. Ejemplo de una restricción en un STI

1.1.5 Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta

Es una aplicación desarrollada como parte de los trabajos de diploma de la Facultad 4 de la Universidad de las Ciencias Informáticas que tiene como objetivo apoyar el aprendizaje del lenguaje estructurado de consulta en la asignatura Sistema de Bases de Datos. El sistema consiste en un ambiente inteligente, que permite realizar la corrección y evaluación de los ejercicios realizados por los estudiantes y a la vez brindar información sobre los errores cometidos.

La herramienta cuenta con una sesión destinada a los administradores que serán los encargados además de otras acciones, de gestionar los usuarios y los problemas. La aplicación dispone 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.

El sistema cuenta con un conjunto de restricciones que se encuentran implementadas de forma estática en el código que no pueden ser modificadas, eliminadas o visualizadas, ni es posible la inserción de una nueva restricción. Además, no brinda la posibilidad al profesor de observar los errores cometidos por los estudiantes lo que dificulta el proceso de enseñanza.

1.2 Análisis de sistemas similares

En el mundo existen varios sistemas para el apoyo a la enseñanza basados en Tutores Inteligentes y Jurados en Línea, por lo que se hace necesario un estudio para analizar soluciones ya existentes que pudieran ser de ayuda al desarrollo del sistema, lo que hace necesario realizar la búsqueda tanto a nivel nacional como internacional. A continuación se muestran los resultados de la investigación:

1.2.1 Nivel internacional

Learning SQL with a computerized Tutor (Aprendiendo SQL con un Tutor computarizado) esta designado como un ambiente de práctica; se supone que los estudiantes hayan recibido conocimientos de los conceptos de gestión de base de datos previamente. El sistema no es un sustituto para el estilo convencional de educación, es un complemento, solamente trabaja sobre la cláusula SELECT del lenguaje SQL y las restricciones del mismo se encuentran implementadas directamente en el núcleo del tutor inteligente (Mitrovic, 1998).

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).

1.2.2 Nivel nacional

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 (Santana 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.

Juez Caribeño en Línea (COJ) es el juez en línea desarrollado en la Universidad de las Ciencias Informáticas para el apoyo a las competencias internas de la misma así como nacionales e internacionales. También es utilizada para el entrenamiento de los equipos destinados a competir en representación de la Universidad y los que representarán al país en competiciones internacionales (FORTES, 2008).

Luego de un análisis de las diferentes herramientas para el apoyo a los estudiantes en la asignatura Sistema de Bases de Datos que sirvan como guía para el desarrollo del componente, el autor considera que aunque las mismas evalúan de forma automática respuestas enviadas por los estudiantes solo son funcionales para la sentencia SELECT del lenguaje SQL, otras no generan retroalimentación a los estudiantes y ninguna cuenta con un gestor automático de restricciones.

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 (Sommerville, 2005). 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 de software

Una Metodología de desarrollo de software es un enfoque estructurado para el desarrollo de software que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (Valdéz, 2014).

Una metodología se basa en una combinación de los modelos de proceso genéricos para obtener como beneficio un software que soluciones un problema. Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades, junto con prácticas, técnicas recomendadas y guías de adaptación de la metodología al proyecto. La complejidad del proceso de creación de software es netamente dependiente de la naturaleza del proyecto mismo (Ramirez, 2015).

Para el desarrollo del sistema propuesto se realiza un estudio de las metodologías ágiles, puesto que las mismas proponen una estrecha relación entre el cliente y el equipo de desarrollo, no existe un contrato tradicional, se aplican a proyectos con requerimientos cambiantes o imprecisos, y están dirigidas fundamentalmente para equipos pequeños; se debe hacer entregas funcionales continuamente y el cliente es parte del equipo de desarrollo (Letelier, 2006).

1.3.1.1 Programación Extrema

La Programación Extrema, o *Extreme Programming* (XP por sus siglas en inglés), es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, lo que promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y además, propicia un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes (Penadés, 2006).

Esta metodología tiene como base la simplicidad y como objetivo principal la satisfacción del cliente; para lograrlo se deben tomar en cuenta cuatro valores fundamentales: Comunicación, Simplicidad, Retroalimentación y Coraje. XP contiene 4 fases fundamentales para llevar a cabo el proceso de desarrollo (Letelier, 2006).

- Exploración
- Planificación
- Iteraciones
- Producción

Herramientas de la XP:

- Historias de usuarios
- Casos de prueba de aceptación
- Tareas de ingeniería
- Tarjetas CRC (Clase-Responsabilidad-Colaboración)

1.3.1.2 Scrum

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Presenta dos características fundamentales. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (Penadés, 2006).

1.3.1.3 AUP

El AUP es una versión simplificada del Rational Unified Process (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones del software de negocio mediante el uso de técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas, Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de la Base de Datos para mejorar la productividad AUP se preocupa especialmente de la gestión de riesgos (Cabrera, 2008).

Propone que aquellos elementos con alto conflicto obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo. Para ello, se crean y mantienen listas que identifican las debilidades desde etapas iniciales del proyecto. Especialmente relevante en este sentido es el desarrollo de prototipos ejecutables durante la base de elaboración del producto, donde se demuestre la validez de la arquitectura para los requisitos clave del producto y que determinan los riesgos técnicos (Cabrera, 2008).

El proceso AUP establece un modelo más simple que el que aparece en RUP por lo que reúne las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño en una sola. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP. Al igual que RUP, en AUP se establecen cuatro fases que transcurren de manera consecutiva: Concepción, Elaboración, Construcción y Transición (Cabrera, 2008).

1.3.1.4 AUP-UCI

Esta metodología surge por la necesidad de la universidad de tener una metodología capaz de adaptarse a los diferentes proyectos que se desarrollan en los centros. La misma mantiene los principales aspectos de AUP. Cuenta con 4 escenarios para el guiar el desarrollo, las características de cada uno de ellos se muestran a continuación.

Escenario No 1: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema. Es necesario que se tenga claro por el proyecto que los CUN muestran como los procesos son llevados a cabo por personas y los activos de la organización.

Escenario No 2: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Escenario No 3: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, lo cual propicia objetividad, solidez, y su continuidad. Se debe tener presente que este escenario es muy conveniente si se desea representar una gran cantidad de niveles de detalles y la relaciones entre los procesos identificados.

Escenario No 4: Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

1.3.1.5 Selección de la metodología a utilizar para el desarrollo del sistema

Para la selección de la metodología de desarrollo se toma como premisa las características del equipo de desarrollo dado que el mismo está compuesto por un solo integrante y que los requisitos del cliente pueden variar con el desarrollo del trabajo por lo que se hace necesario el uso de una metodología ágil, pero a su vez se necesita generar documentación.

Una vez realizado el análisis de las diferentes metodologías, a pesar de que todas pueden ser utilizadas para guiar el desarrollo del componente se selecciona **AUP-UCI** dado que el cliente exigió la utilización de la misma.

1.3.2 Framework de Desarrollo

Un framework simplifica el desarrollo de una aplicación mediante la automatización de patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, fuerza al desarrollador a crear código más legible y más fácil de mantener. Por último, un

framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas (Potencier, 2008).

Symfony es un framework diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones web. Separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, brindándole al desarrollador la posibilidad de dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web (Potencier, 2008).

Está desarrollado completamente con PHP 5. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows (Potencier, 2008).

Características de Symfony

A continuación se enuncian las características de Symfony según (Potencier, 2008):

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares)
- Independiente del sistema gestor de bases de datos
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros

1.3.3 Lenguajes del lado del servidor

PHP: Es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Gran número de páginas y portales web están creadas con PHP (Prior,

2015). Es el acrónimo de Hipertext Preprocesor. El mismo es utilizado del lado del servidor, es independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la página PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores (Alvarez, 2001).

Ventajas

- Gratuito
- Independencia de plataforma.
- Rapidez
- Seguridad
- Compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC.

La versión a utilizar es la 5.4 por el conocimiento del desarrollador en el trabajo con la misma y porque el servidor web que será utilizado trabaja sobre dicha versión.

1.3.4 Lenguajes del lado del cliente

JavaScript: Es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios (Gallego, 2016).

La versión a utilizar es la 1.8 porque es la utilizada en la aplicación a la cual se integrará el componente.

HTML: Se utiliza para el desarrollo de páginas de Internet. Se trata de la sigla que corresponde a HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es decir, lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto. Se trata de un formato abierto que surgió a partir de las etiquetas SGML (Standard Generalized Markup Language). Concepto traducido generalmente como “Estándar de Lenguaje de Marcado

Generalizado” y que se entiende como un sistema que permite ordenar y etiquetar diversos documentos dentro de una lista (Sánchez, 2015).

Este lenguaje es el que se utiliza para especificar los nombres de las etiquetas que se utilizarán al ordenar, no existen reglas para dicha organización, por eso se dice que es un sistema de formato abierto. Es un lenguaje muy simple y general que sirve para definir otros lenguajes que tienen que ver con el formato de los documentos. El texto en él se crea a partir de etiquetas, también llamadas tags, que permiten interconectar diversos conceptos y formatos (Sánchez, 2015).

Destacar que el HTML permite ciertos códigos que se conocen como scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP (Sánchez, 2015).

Para el desarrollo del componente se decide utilizar HTML en su versión 5 porque la aplicación a la cual se integrará fue desarrollada mediante el uso de esta versión.

Bootstrap: Es un framework que ofrece un diseño adaptativo, también conocido como Responsive Design. A pesar de la diversidad de elementos que aporta Bootstrap, se puede considerar que sus diseños son simples, limpios y rápidos; por lo que se ofrece una gran agilidad a la hora de cargarse en los navegadores y adaptarlos al tamaño de cada dispositivo. Combina el uso de JavaScript con plantillas CSS3 para ofrecer diversos elementos tales como Botones, Menús desplegables, Formularios incluyendo todos sus elementos e integración jQuery para ofrecer ventanas y tooltips dinámicos (Sánchez, 2015).

La versión a utilizar es la 3.0 porque la misma es la que fue utilizada para el desarrollo de la aplicación a la cual se integrará el componente y por el conocimiento sobre el uso de la misma.

CSS: Las hojas de estilo en cascada (Cascading Style Sheets, CSS), permiten desarrollar la creatividad en el diseño con una intuición sin precedentes. Las CSS constituyen un mecanismo para asociar estilos de composición a documentos estructurados, del tipo HTML o XML. Aplicables a cualquier navegador, admiten un mayor control sobre los distintos elementos de una página, lo que permite definir el estilo de las fuentes, el color, el espaciado del texto, la posición del contenido, e incluso variaciones en el sonido en los elementos auditivos. Estos estilos pueden definirse para luego ser aplicados al código de cualquier documento (Patterson, 2003).

Para el desarrollo del componente se utilizará CSS en su versión 3.0 por la experiencia del desarrollar y porque fue la utilizada en el desarrollo del sistema al cual se integrará el componente mediante el uso del framework Bootstrap.

JQuery: Es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

JQuery es software libre y de código abierto. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. (Suárez, 2015).

La versión a utilizar es la 1.10.2 porque la misma es la utilizada en la implementación a la cual se integrará el componente.

1.3.5 Lenguaje de Modelado

El lenguaje modelado unificado (UML): es un lenguaje gráfico que sirve para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Pumarejo, 2016). La capacidad de UML para modelar cualquier sistema de software y el hecho de que haya sido adoptado por el OMG[1] 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.

1.3.6 Herramienta CASE para el Modelado UML

Visual Paradigm for UML: es una herramienta de modelado desarrollada para asistir el proceso de Ingeniería de Software, este se encuentra basado en UML y soporta el ciclo de vida completo del desarrollo de software, además cuenta con funcionalidades más avanzadas que las presentes en el Rational Rose, lo que permite agilizar considerablemente el trabajo. Sus principales características son las siguientes: presenta licencia gratuita y comercial, soporta aplicaciones web, disponible en varios idiomas, fácil de instalar y actualizar, compatible entre versiones, entorno gráfico amigable para el usuario y disponible en múltiples plataformas (Windows/Linux/Mac OS X), permite diseñar de todo tipo de diagramas y generar documentación (Núñez, 2016).

Para el desarrollo del sistema se utilizará la versión 8.0 por no ser de carácter privado, la experiencia y conocimiento previos en el uso de la herramienta.

1.3.7 Servidor de Base de Datos

PostgreSQL: es un potente motor de bases de datos orientadas a objetos y libre, que tiene prestaciones y funcionalidades equivalentes a muchos gestores de bases de datos comerciales. Es

[1] **OMG:** Grupo de Gestión de Objetos

más completo que MySQL ya que permite métodos almacenados, restricciones de integridad, vistas, etc (Colorado, 2015).

Se basa en el uso de un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará en funcionamiento (Ma, 2010).

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.

1.3.8 Entorno de Desarrollo Integrado

phpStorm: es un entorno de desarrollo integrado (IDE) especialmente destinado para los desarrolladores web que necesitan las herramientas adecuadas para editar PHP, HTML, CSS, JavaScript y XML, trabajar con VCS, SQL, además añade soporte avanzado para otras herramientas específicas para el desarrollo web (JetBrains, 2016).

La versión a utilizar es la 8.0 porque la misma no es de carácter privativo, brinda soporte a los lenguajes a utilizar en el desarrollo de la aplicación así como el soporte que brinda al framework Symfony y el conocimiento previo sobre el uso de la misma.

1.3.9 Servidor Web

Un servidor web es un programa que atiende y responde a las diversas peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la versión segura, cifrada y autenticada de HTTP). Un servidor web básico tiene un esquema de funcionamiento muy sencillo, ejecuta de forma infinita el bucle siguiente (Mateu, 2004):

1 Espera peticiones en el puerto TCP asignado (el estándar para HTTP es el 80)
2 Recibe una petición
3 Busca el recurso en la cadena de petición
4 Envía el recurso por la misma conexión por donde ha recibido la petición
5 Vuelve al punto 2

Apache: es un servidor web de código libre, robusto, cuya implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo (Apache Group) que, mediante el uso de Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada". Además, se caracteriza por ser un servidor web que posee gran

rapidez, eficiencia y flexibilidad y se adapta a los nuevos protocolos HTTP. Todo ello se une al hecho de ser multiplataforma, adaptable a diferentes entornos y necesidades y extensible a partir de su condición de modularidad (Lara, 2016).

La selección de Apache como servidor web estuvo determinada porque el mismo está respaldado por una comunidad de usuarios numerosa, por lo que existe una documentación extensa del mismo disponible en Internet, además del conocimiento previo del autor sobre el trabajo con dicho servidor web. La versión a utilizar es la 2.4.10.

1.3.10 Mapeador de Objetos Relaciones

Doctrine ORM: es una capa de abstracción de datos y mapeador de objetos relacionales (ORM en sus siglas en ingles) que permite independiente de la base de datos (MySQL, PostgreSQL, Oracle, etc.) cargar los datos del esquema relacional y las tuplas de una tabla y representarlos en objetos propios del lenguaje de programación huésped del servidor tal como lo es PHP, además de representar los datos en objetos, también permiten editar y agregar datos a estos objetos, pudiéndose después guardarse estos datos en la base de datos. En el caso del sistema, la abstracción de la base de datos se hará con el sistema manejador de base de datos PostgreSQL (Carrillo, 2015).

1.4 Conclusiones del capítulo

Teniendo en cuenta los métodos científicos y las técnicas de recopilación de datos empleadas en la presente investigación, se pudo arribar a las siguientes conclusiones:

- El análisis del estado del arte asociado al objeto de estudio, permitió identificar los conceptos principales para fundamentar teóricamente la propuesta de solución.
- No hay evidencia de Sistemas Tutore Inteligente que utilicen un Sitema de gestión para el trabajo con las restricciones.
- Aunque el estudio de los sistemas similares permitió conocer que estos no satisfacen las necesidades y condiciones requeridas, el análisis realizado a sus características servirá de base para identificar las posibles funcionalidades de la propuesta de solución.
- Teniendo en cuenta las características del AIA-SQL se percibe la necesidad de la inclusión de un Sistema para gestionar las restricciones y la generación de retroalimentación.
- El análisis comparativo establecido entre las metodologías, lenguajes de programación, herramientas y tecnologías más usadas en el desarrollo de sitios web, permitió seleccionar las más apropiadas para guiar y desarrollar la solución del problema planteado.

Capítulo II: Propuesta de solución

2 Introducción

En el proceso de construcción de software una de las partes más difíciles del trabajo conceptual es decidir precisamente qué construir, de modo que se logre entender las necesidades del cliente y los demás involucrados. En el presente capítulo se hace una descripción de la propuesta de solución a la situación problemática planteada. Se especifican los requisitos funcionales y no funcionales y se obtienen como artefactos fundamentales las historias de usuarios.

2.1 Propuesta de solución

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 componente a desarrollar tiene como propósito incorporar al Ambiente Inteligente de Aprendizaje del Lenguaje Estructurado de Consulta (AIA-SQL) la posibilidad de añadir un sistema de gestión de restricciones que permitirá a los profesores insertar, eliminar, editar y visualizar las mismas sin necesidad de ir al código fuente de la aplicación.

El conjunto de restricciones presentes en el sistema será incrementado en la propuesta de solución debido a que la misma cuentas solo con 7, esto permitirá que la evaluación de las soluciones enviadas por los estudiantes sea más rigurosa. También permitirá la generación de una retroalimentación más completa.

Con el desarrollo e integración del nuevo componente se pretende optimizar el funcionamiento de la versión anterior, permitir la gestión dinámica de las restricciones para la evaluación del código SQL así como crear un sistema de retroalimentación que les permita a los profesores dar seguimiento a los resultados de los estudiantes.

2.2 Modelo de dominio

Un modelo del dominio es una representación de las clases conceptuales del mundo real, no de componentes software. No se trata de un conjunto de diagramas que describen clases software, u objetos software con responsabilidades (Larman, 2003).

Es posible tomar como punto de inicio del proyecto al modelo de dominio para el diseño del sistema. Cuando se programa orientado a objetos el sistema de forma interna imita una realidad en cierta medida, por lo cual el mapa de conceptos del modelo de dominio representa una primera versión del sistema (Larman, 2003).

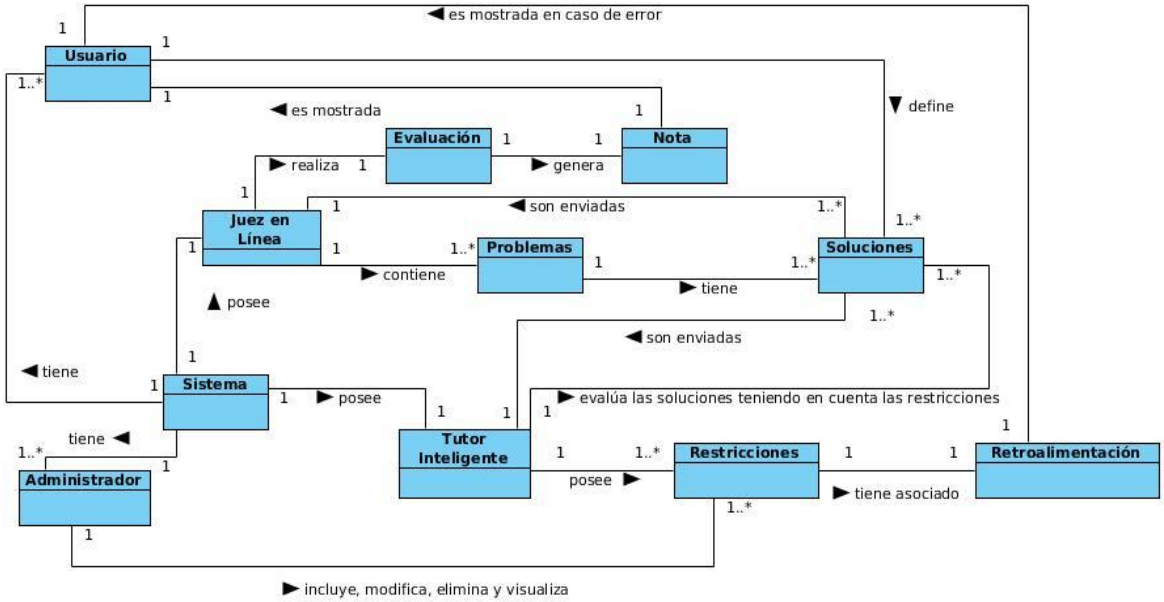

Imagen 3. Modelo de dominio

2.2.1 Definición de las clases del Modelo de Dominio

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

Restricciones: Condiciones por las cuales serán evaluadas las respuestas de los estudiantes. Estas están formadas por una condición de relevancia, una condición de satisfacción y un valor de orden que permitirá evaluarla en sentido lineal o inverso.

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, las restricciones, 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 mediante la evaluación las soluciones enviadas por los estudiantes.

Retroalimentación: Información en forma de ayuda que se proporciona al estudiante al evaluar las restricciones presentes en el sistema.

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

Nota: 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 enviadas por los estudiantes y administradores.

Juez en Línea: Encargado de la evaluación de los problemas del sistema.

Tutor Inteligente: Identifica los errores cometidos por los estudiantes mediante el conjunto de restricciones y genera una retroalimentación.

2.3 Especificación de requisitos funcionales

El levantamiento de requisitos de software constituye una de las etapas más importantes en el proceso de ingeniería de requisitos. Durante esta etapa se realiza un trabajo en conjunto entre los participantes y los desarrolladores con el objetivo de identificar el problema y especificar un conjunto preliminar de requisitos para la solución (Pressman, 2005)

2.3.1 Requisitos funcionales

Existe un consenso general en la ingeniería del software acerca del significado de requisitos funcionales. Estos suelen definirse como:

- Una función que el sistema tiene que ser capaz de ejecutar.
- Lo que el producto (software) debe hacer.
- Lo que el sistema debería hacer.

Todas estas definiciones contemplan implícitamente aspectos operacionales del software (función sistema) (Castro, 2014). Una vez estudiado el estado del negocio, se identificaron como requisitos funcionales los siguientes:

RF 1 Ver restricción: El sistema debe permitir a los usuarios con rol Administrador visualizar los datos de las restricciones del sistema.

El sistema debe permitir al usuario ver los siguientes datos:

- Autor
- Tipo
- Relevancia
- Satisfacción
- Orden

RF 2 Editar restricción: El sistema debe permitir a los usuarios con rol Administrador modificar las restricciones existentes en el sistema al seleccionar la opción:

- Editar

El sistema debe permitir modificar los siguientes datos:

- (*) Tipo
- (*) Relevancia
- (*) Satisfacción
- (*) Orden

El sistema debe permitir modificar todos los datos de una restricción, excepto el autor.

RF 3 Eliminar restricción: El sistema debe permitir a los usuarios con rol Administrador eliminar una o varias restricciones en el sistema.

RF 4 Insertar restricción: El sistema debe permitir a los usuarios con rol Administrador insertar nuevas restricciones en el sistema al solicitar los siguientes datos:

- (*) Tipo
- (*) Relevancia
- (*) Satisfacción
- (*) Orden

Además se guarda el siguiente dato en la entidad:

- Autor

RF 5 Evaluar restricción: El sistema realiza la evaluación de las restricciones presentes en el sistema de forma invisible a los usuarios para comprobar si las respuestas de los estudiantes están correctas.

RF 6 Generar retroalimentación: El sistema debe realizar la evaluación de las respuestas de los estudiantes mediante el uso de las restricciones presentes en el sistema. Si al evaluar las soluciones encuentra un error el sistema debe ser capaz de generar una retroalimentación, la misma se muestra al estudiante y se guarda en la base de datos.

RF 7 Ver retroalimentación: El sistema debe permitir a los usuarios con rol Estudiante visualizar la retroalimentación asociada al error cometido en la solución enviada.

RF 8 Listar restricciones: El sistema debe mostrar a los usuarios con rol Administrador una lista con todas las restricciones que contiene el sistema al visualizar los siguientes datos:

- Autor

- Tipo
- Relevancia
- Satisfacción
- Orden

RF 9 Listar retroalimentaciones generadas: El sistema debe mostrar a los usuarios con rol Administrador o Estudiante una lista con las retroalimentaciones generadas por el sistema.
El sistema debe permitir al usuario con rol Administrador ver los siguientes datos:

- Mensaje
- Problema
- Usuario
- Ver Solución

2.3.2 Requisitos no funcionales

Un requisito no funcional especifica restricciones físicas con uno funcional, tales como restricciones del entorno o de implementación, tiempos de respuesta, dependencias de la plataforma, mantenibilidad, extensibilidad y fiabilidad (Castro, 2014). Para el desarrollo de la propuesta de solución se definieron los siguientes requisitos no funcionales:

Confiabilidad

RFN 1: Solo podrán acceder a las restricciones los usuarios previamente registrados con rol Administrador.

RNF 2: Antes de realizar alguna acción irreversible el componente debe mostrar alguna advertencia, ejemplo cuando se desea borrar una restricción.

Usabilidad

RNF 3: El componente tiene como objetivo permitir el trabajo dinámico con las restricciones y generar una retroalimentación por los errores cometidos para ayudar a la comprensión del contenido.

Restricciones de diseño e implementación

RNF 4: 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.

RNF 5: El sistema debe ser implementado mediante el uso del IDE phpStorm en su versión 8.0.

RNF 6: El marco de trabajo que se debe utilizar para el desarrollo es el framework Symfony en su versión 2.7.

RNF 7: Se debe utilizar para el desarrollo, como lenguaje de programación del lado del servidor PHP en su versión 5.4.12.

RNF 8: 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 mediante el uso de la librería JQuery.

RNF 9: 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

RNF 10: El componente debe mantener el diseño utilizado para desarrollar el sistema actual.

RNF 11: El componente debe notificar en caso de existir un error.

2.3.3 Descripción de los actores del sistema

Para el desarrollo del componente se mantienen los roles asociados a cada usuario definidos en el sistema actual para controlar el acceso a las nuevas funcionalidades con que contará la aplicación. A continuación se muestra roles definidos y su descripción.

Rol	Descripción
Administrador	Encargado de gestionar los usuarios que tendrán acceso al sistema así como los problemas que en este se incluyan. Se encargará de suministrarle solución a los problemas creados la cual será utilizada por el sistema para evaluar la solución enviada por los estudiantes. Gestionará las restricciones presentes en el sistema así como la retroalimentación. Configurar y ver perfil.
Estudiante	Encargado de brindar solución a los problemas, recibir evaluación y retroalimentación. Además tiene la posibilidad realizar las mismas acciones que el usuario del sistema. Configurar y ver perfil.
Invitado	Encargado de realizar acciones en el sistema sin necesidad de autenticación (ver los problemas, ver los envíos, ver las estadísticas, filtrar problemas, buscar usuarios).
Sistema	Encargado de realizar todas las acciones referentes a la evaluación y la retroalimentación de forma automática e invisible a los usuarios.

Tabla 1. Descripción de los actores del sistema

2.3.4 Historias de usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento del proyecto las historias de usuario pueden romperse en varias historias de usuario, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en solo unas semanas (Goggi, 2012). Para documentar las HU se utiliza una plantilla a cual define los aspectos que se muestran a continuación:

- Número: posee el número asignado a la HU
- Nombre del requisito: atributo que contiene el nombre del requisito funcional al que corresponde la HU
- Programador: usuario encargado de desarrollar la HU
- Prioridad: evidencia el nivel de prioridad de la HU en el negocio
- Tiempo estimado: estimación hecha por el equipo de desarrollo del tiempo de duración de la HU
- Tiempo real: tiempo real de desarrollo de la HU
- Descripción: Descripción de la HU
- Observaciones: Dependencias de la HU

A continuación se muestran HU asociadas a los requisitos funcionales Evaluar restricciones, Insertar restricción y Listar restricciones. Las restantes se encuentran en el Anexo #1.

Número: 5	**Nombre del requisito:** Evaluar restricción
Programador: Alexander Cutiño Rivera	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**21 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permitir evaluar las restricciones en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):**	

Para evaluar las restricciones tiene que: -Existir al menos una restricción en el sistema. **3- Flujo de la acción a realizar:** - El sistema debe evaluar las restricciones en las soluciones introducidas por los estudiantes y generan un mensaje en caso que el estudiante haya dado una respuesta errónea.
Observaciones: Las restricciones son para evaluar las respuestas de los estudiantes.
Prototipo de interfaz: N/A

Tabla 2. Historia de usuario No. 5 Evaluar restricciones

Número: 4	**Nombre del requisito:** Insertar restricción
Programador: Alexander Cutiño Rivera	**Iteración Asignada:** 1era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días
Descripción: **1- Objetivo:** Permitir insertar una restricción en el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** Para insertar una restricción hay que: -Tener en cuenta los siguientes datos: tipo, condición de satisfacción, condición de relevancia y orden. - Estar autenticado en el sistema con el rol Profesor o Administrador. **3- Comportamientos válidos y no válidos (flujo central y alternos):** Los campos tipo, condición de satisfacción, condición de relevancia y orden son obligatorios. Tipo: Campo que permite seleccionar el tipo de consulta SQL de la restricción. Condición de satisfacción: Los valores del campo tienen que ser tokens del lenguaje plpgsql y ser tokens del tipo de sentencia especificado. Condición de relevancia: Los valores del campo tienen que ser tokens del lenguaje	

plpgsql y ser tokens del tipo de sentencia especificado o la palabra “param”.

Orden: Campo que permite seleccionar el orden de revisión consulta SQL de la restricción.

4- Flujo de la acción a realizar:

- El sistema debe permitir incluir y/o seleccionar los datos para incluir una nueva restricción.
- Cuando el usuario incluye y/o selecciona correctamente los datos necesarios para incluir una restricción y selecciona la opción Insertar, se crea un nuevo elemento y el sistema muestra la restricción insertada.
- Si los datos están incompletos o incorrectos se mostrará un error y se brinda la posibilidad al usuario de realizar nuevamente la acción en cuestión.
- Si selecciona la opción Volver a la lista regresará a al listado de restricciones.

Observaciones: Las restricciones son para evaluar las respuestas de los estudiantes.

Prototipo de interfaz:

Tabla 3. Historia de usuario No. 4 Insertar restricciones

Número: 8	**Nombre del requisito:** Listar restricciones
Programador: Alexander Cutiño Rivera	**Iteración Asignada:** 3era
Prioridad: Alta	**Tiempo Estimado:**3 días
Riesgo en Desarrollo: N/A	**Tiempo Real:** 2 días

Descripción:

1- Objetivo:

Permitir listar los datos de las restricciones en el sistema.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para listar restricciones hay que:

- Estar autenticado en el sistema con el rol Administrador.
- Debe existir en el sistema al menos una restricción.

3- Flujo de la acción a realizar:

Cuando el usuario selecciona la opción Restricciones el sistema muestra el listado de restricciones insertadas en el sistema.

El sistema muestra los siguientes datos de las restricciones:

- Tipo
- Relevancia
- Satisfacción
- Orden

Además, el usuario tiene la posibilidad de ver detalles de una restricción, editarla y eliminarla.

Observaciones: Las restricciones son para evaluar las respuestas de los estudiantes.

Prototipo de interfaz:

Tabla 4. Historia de usuario No. 8 Listar restricciones

2.4 Conclusiones del capítulo

Con el desarrollo de este capítulo se obtuvieron los siguientes resultados:

- Fue descrita la propuesta de solución que servirá como punto de partida la realización del análisis y diseño y fueron definidos los principales conceptos de la misma, los cuales quedaron representados en el modelo de dominio, el cual brindó una mejor comprensión de la solución.
- Fueron identificados los actores que interactúan con el sistema siendo estos Administrador, Estudiante, Invitado y Sistema.
- De la propuesta de solución plateada se detectaron un total de 9 requisitos funcionales y 11 no funcionales evidenciados en 9 Historias de usuario.

Capítulo III: Análisis, diseño, implementación y prueba

3 Introducción

En el presente capítulo se realiza el análisis y diseño del componente como parte del flujo de trabajo que propone AUP-UCI. Se generan los artefactos correspondientes al modelo del análisis y el modelo del diseño, además se diseña el modelo de base de datos y el modelo de despliegue. También a partir de los artefactos generados en el análisis y diseño se desarrolla la implementación, donde se generan las clases pertenecientes al componente que se integrará al sistema. También se presenta una descripción detallada de a través de los diagramas de componentes del proceso de implementación de las funcionalidades y se aplicarán las pruebas con el objetivo de verificar la calidad del producto y la aceptación del mismo.

3.1 Modelo del 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 mediante la utilización del 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 Diagramas de clases del análisis (DCA)

Una clase de análisis representa una abstracción de una o varias clases, las mismas se ajustan 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, así mismo muestran una estructura de datos lógica y contribuyen a comprender de qué información depende el sistema (Jacobson, 2000).

A continuación se muestran los diagramas de clases del análisis de los requisitos Evaluar restricciones, Ver retroalimentación y Listar restricciones, en el Anexo #2 se encuentran los demás diagramas de clases del análisis generados.

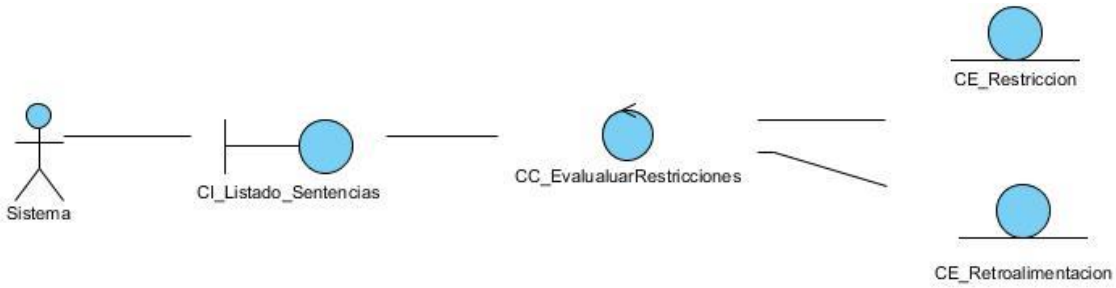

Imagen 4. DCA Evaluar restricciones.

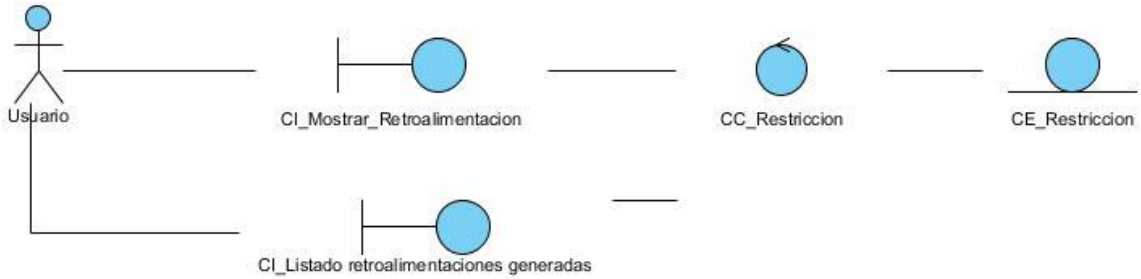

Imagen 5.DCA Ver retroalimentación.

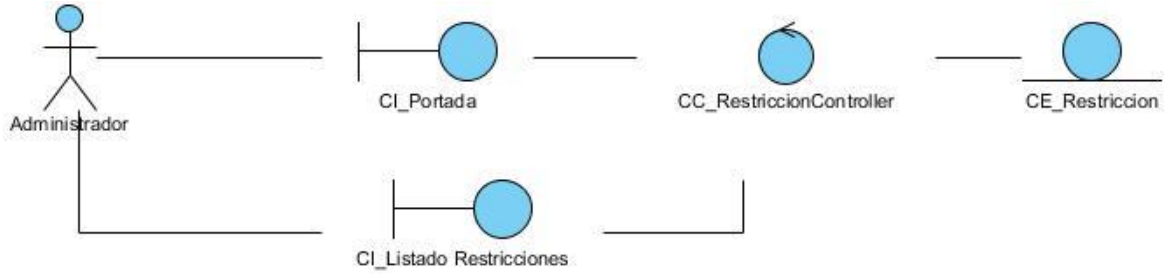

Imagen 6. DCA Listar restricciones.

3.1.2 Diagramas 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, se visualiza cómo interactúan los objetos secuencialmente o en paralelo y se enumeran los mensajes que se envían unos a otros.

A continuación se muestran los diagramas de colaboración del análisis de los requisitos Ver restricción, Insertar restricción y Listar restricciones, los restantes diagramas se encuentran en el Anexo #3.

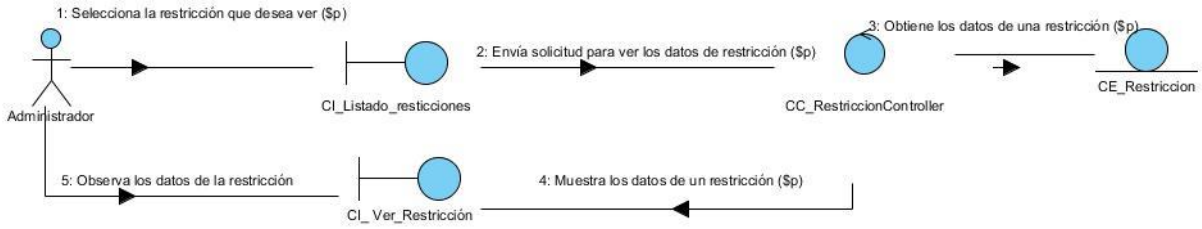

Imagen 7 DCO Ver restricción.

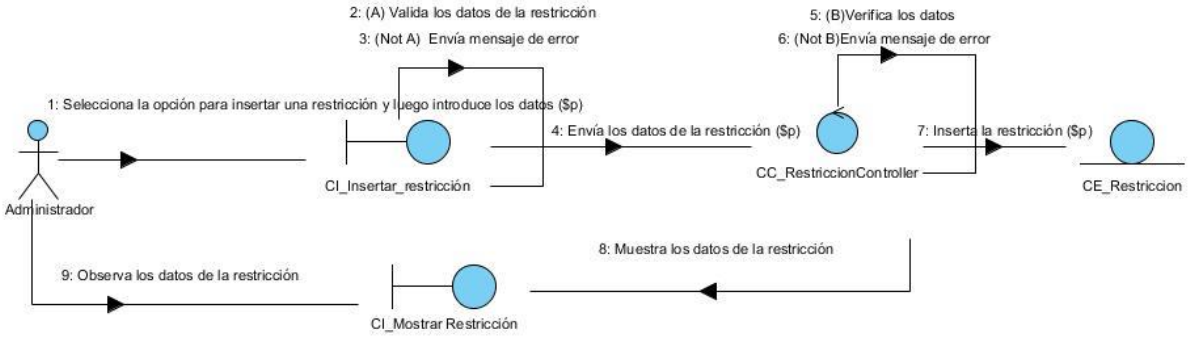

Imagen 8.DCO Insertar restricción.

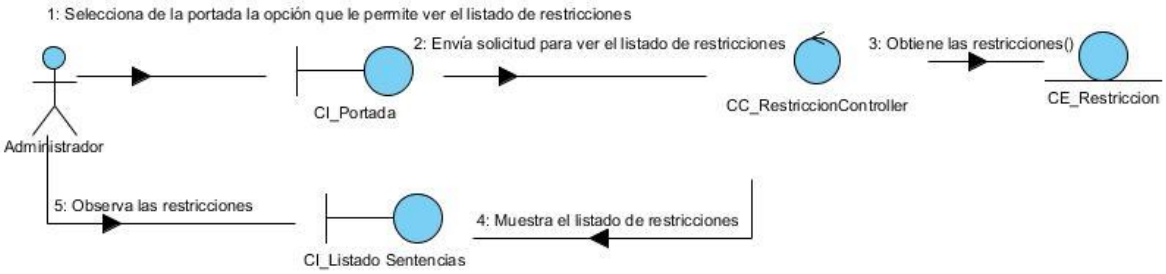

Imagen 9. DCO Listar restricciones.

3.2 Patrón arquitectónico

Los patrones de arquitectura de software constituyen una vía en la solución de problemas de arquitectura de software. Poseen un nivel de abstracción mucho mayor que los patrones de diseño. Además brindan una descripción de los elementos y el tipo de relación que tienen, así como las restricciones para su uso. Los patrones se especifican mediante la descripción de los componentes, con sus responsabilidades, relaciones, y las formas en que colaboran (Larman, 1999).

Como base para el desarrollo de la aplicación propuesta se utilizó el marco de trabajo Symfony dado que fue el utilizado para la primera versión de la aplicación, el mismo 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 y se facilita de esta forma la evolución por separado de ambos aspectos lo que incrementa 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.

- ***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 mediante el procesamiento de toda la información necesaria y realiza los cambios apropiados en el modelo o en la vista. Está representado por las clases *Controller*.

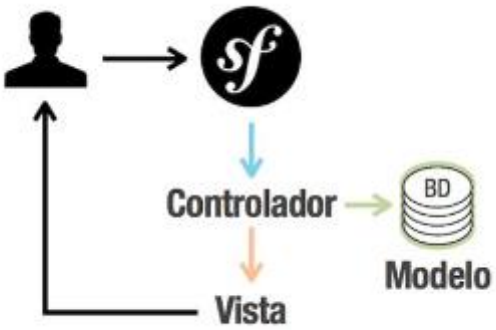

Imagen 10. Patrón arquitectónico de Symfony

A continuación se evidencia el uso de esta arquitectura en el componente:

Clases controladoras

Ruta: AIA-SQL/Juez/src/JO/RestriccionBundle/Controller

- EvaluadorRestricciones.php
- RestriccionController.php
- RetroalimentacionController.php

Clases de la vista

Ruta: AIA-SQL/Juez/src/JO/RestriccionBundle/Resources/views

- Restricción

  Ruta: AIA-SQL/Juez/src/JO/RestriccionBundle/Resources/views/Restriccion

  - edit.html.twig
  - index.html.twig
  - show.html.twig
  - new.html.twig
- Retroalimentación

  Ruta: AIA-SQL/Juez/src/JO/RestriccionBundle/Resources/views/Retroalimentacion

  - index.html.twig
  - show.html.twig

Clases del modelo

Ruta: AIA-SQL/Juez/src/JO/RestriccionBundle/Entiy

- Restriccion.php
- Retroalimentacion.php

3.3 Patrones de diseño

Los patrones de diseño son soluciones acertadas a un problema general, que establecen pautas para definir estructuras de diseño en el desarrollo de un software. Estos, están más enfocados a los diseños orientados a objetos pero según dichos patrones a menudo tienen en cuenta características de los objetos tales como la herencia y el polimorfismo que proporcionan generalidad. (Sommerville, 2005). En el desarrollo de la aplicación a la que se incorporará el componente se utilizó Symfony como marco de trabajo, el mismo hace uso de un conjunto de patrones de diseño en su

implementación. A continuación se describen los patrones de diseño utilizados en la implementación 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". 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 mediante el uso de la función get(). A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```
protected function execute(InputInterface $input, OutputInterface $output)
{
    $em = $this->getContainer()->get('doctrine')->getManager();
    // Hacemos lo que sea

    $soluciones = $em->getRepository('SolucionBundle:Solucion')->findBy(array('estado' => 'Pendiente'), array('fecha' => 'ASC'));

    $evaluados = 0;
    foreach ($soluciones as $solucion) {
        $solucion->setEstado('Semantica');

        $usuario = $em->getRepository('UsuarioBundle:Usuario')->find($solucion->getUsuario());
        $problema = $em->getRepository('ProblemaBundle:Problema')->find($solucion->getProblema());

        $solucion_estudiante = $solucion->getSolucion();
        $tipo_sql = $problema->getTipoSql();

        $semantica = new EvaluadorRestricciones($solucion_estudiante, $tipo_sql, $em);
```

Imagen 11. Ejemplo del uso del patrón 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) (Potencier, 2008).

Patrón Iterator (Iterador)

El patrón de diseño Iterator proporciona un mecanismo de acceso secuencial a los objetos almacenados en una colección. Este patrón permite entre otras ventajas que se pueda iterar en cualquier orden simplemente aplicando un algoritmo genérico de ordenación a la colección (Marco, 2013). Este patrón es el que se utiliza para a dar solución a al trabajo con la colección de restricciones, a continuación se muestra la evidencia de su utilización:

```php
private function analizarSentencia($tipoSentencia, $sentencia)
{
    $mensaje = null;
    for ($i = 0; $i < count($sentencia); $i++) {
        $banderaVerificarPosicion = false;
        $tokenActual = $sentencia[$i];
        $listaRestricciones = $this->obtenerRestricciones($tipoSentencia);      //guardo las restricciones que responden al tipo de sentencia
        $palabraRelevancia = null;
        $palabraSatisfaccion = null;

        foreach ($listaRestricciones as $restricciones) {
            //si el token actual es una palabra que responde a una condición de relevancia compruebo si sus condiciones de satisfacción se cu
            if ($restricciones->getPrioridad() == '1') {
                if ($tokenActual == $restricciones->getRelevancia()) {
                    if ($restricciones->getSatisfaccion() == 'param') {
                        $respuesta = $this->verificarParametros($tokenActual, $tipoSentencia);      //si la condicion de satisfacción es para
                        if ($respuesta == true) {
                            $mensaje .= " Después de la palabra reservada '" . strtoupper($restricciones->getRelevancia()) . "' tienen que ve
                        }
```

Imagen 12. Ejemplo del uso del patrón Iterator.

3.3.1 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones (Astudillo, 2012).A continuación se muestran los patrones GRASP utilizados en el desarrollo del sistema:

Experto: Define el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Con la utilización de este patrón se definió dónde colocar en cada clase las funcionalidades que necesitan de esa información. En el sistema se puede observar el uso de este patrón en las clases entidades, ejemplo la clase restriccion.php que es la encargada de conocer toda la información referente a las restricciones. A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```php
102     /**
103      * Set autor
104      *
105      * @param \JO\UsuarioBundle\Entity\Usuario $autor
106      *
107      * @return Restriccion
108      */
109     public function setAutor(\JO\UsuarioBundle\Entity\Usuario $autor = null)
110     {
111         $this->autor = $autor;
112
113         return $this;
114     }
115
116     /**
117      * Get autor
118      *
119      * @return \JO\UsuarioBundle\Entity\Usuario
120      */
121     public function getAutor()
122     {
123         return $this->autor;
124     }
```

Imagen 13. Ejemplo del uso del patrón Experto.

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, lo que permite 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 el sistema se pueden encontrar ejemplos del uso de este patrón en la ubicación *Juez/src/JO/RestriccionBundle/Controller.* A continuación se muestra un ejemplo del uso de este patrón durante el desarrollo de la solución:

```
176     public function updateAction(Request $request, $id)
177     {
178         $em = $this->getDoctrine()->getManager();
179 
180         $entity = $em->getRepository('RestriccionBundle:Restriccion')->find($id);
181 
182         if (!$entity) {
183             throw $this->createNotFoundException('Unable to find Restriccion entity.');
184         }
185 
186         $deleteForm = $this->createDeleteForm($id);
187         $editForm = $this->createEditForm($entity);
188         $editForm->handleRequest($request);
189 
190         if ($editForm->isValid()) {
191             $em->flush();
192 
193             return $this->redirect($this->generateUrl('restriccion_edit', array('id' => $id)));
194         }
195 
196         return $this->render('RestriccionBundle:Restriccion:edit.html.twig', array(
197             'entity' => $entity,
198             'edit_form' => $editForm->createView(),
199             'delete_form' => $deleteForm->createView(),
200         ));
201     }
```

Imagen 14. Ejemplo del uso del patrón Controlador.

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). En el sistema es utilizado en las clases controladoras en las cuales se crean objetos de las clases entidades para el trabajo con ellas. A continuación se muestran ejemplos del uso de este patrón en la investigación:

```
51          $semantica = new EvaluadorRestricciones($solucion_estudiante, $tipo_sql, $em);
```

Imagen 15. Ejemplo del uso del patrón Creador.

```
60          $evaluador = new Evaluador($semantica->getSql(), $solucion_autor, $sql_comprobacion, $tipo_sql, $script);
```

Imagen 16. Ejemplo del uso del patrón Creador.

Alta cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. La información que almacena una clase debe ser coherente y estar en la mayor medida relacionada con la clase. Estas clases mejoran la claridad y la facilidad con que se entiende el diseño. El sistema hace presente este patrón en las clases controladoras dado que las mismas contienen varias funcionalidades que están estrechamente relacionadas estrechamente relacionadas entre sí y son las encargadas de definir las acciones y colaborar con otras para realizar las tareas que las implican a todas. A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```
    public function deleteAction(Request $request)
    {
        $id = $request->get('id');
        $em = $this->getDoctrine()->getManager();
        $entity = $em->getRepository('RestriccionBundle:Restriccion')->find($id);

        $em->remove($entity);
        $em->flush();

        return $this->redirect($this->generateUrl('restriccion'));
    }

    /**
     * Creates a form to delete a Restriccion entity by id.
     *
     * @param mixed $id The entity id
     *
     * @return \Symfony\Component\Form\Form The form
     */
    private function createDeleteForm($id)
    {
        return $this->createFormBuilder()
            ->setAction($this->generateUrl('restriccion_delete', array('id' => $id)))
            ->setMethod('DELETE')
            ->add('submit', 'submit', array('label' => 'Delete'))
            ->getForm();
    }
}
```

Imagen 17. Ejemplo del uso del patrón Alta Cohesión.

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 y se potencie 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. A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```php
/**
 * Set restriccion
 *
 * @param \JO\RestriccionBundle\Entity\Restriccion $restriccion
 * @return Retroalimentacion
 */
public function setRestriccion(\JO\RestriccionBundle\Entity\Restriccion $restriccion = null)
{
    $this->restriccion = $restriccion;

    return $this;
}

/**
 * Get restriccion
 *
 * @return \JO\RestriccionBundle\Entity\Restriccion
 */
public function getRestriccion()
{
    return $this->restriccion;
}
```

Imagen 18. Ejemplo del uso del patrón Bajo Acoplamiento.

3.3.2 Patrones GoF

Describen 23 patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño. Se clasifican en tres categorías basadas en su propósito: ***creacionales***, ***estructurales*** y de ***comportamiento*** (Larman, 2003). En el desarrollo del sistema se hace uso de un patrón de categoría ***estructurales***, los cuales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. A continuación se explica el uso de uno de estos patrones:

Decorator (Decorador): Es un patrón de tipo estructura, ya que permite determinar que clases y objetos serán utilizados para componer estructuras de mayor tamaño (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, esta plantilla es la contenedora de la estructura y el diseño básico de las vistas. A continuación se muestra como se extiende en las diferentes vistas del componente la plantilla “base.html.twig”:

```
{% extends '::base.html.twig' %}
{% block title %}{{ 'menu.admin.Rrrestrictions' | trans }}{% endblock %}
{% block contenido %}
```

Imagen 19. Ejemplo del uso del patrón Decorator.

3.4 Diagrama de clases del diseño (DCD)

Los diagramas de clases del diseño representan la vista estática del sistema y modelan los conceptos asociados al dominio de la aplicación así como los conceptos internos definidos para la parte de la implementación. No se describe el comportamiento del sistema dependiente del tiempo y se representa mediante clases y sus relaciones (Jacobson, 2000).

A continuación se muestran los diagramas de clases del diseño de los requisitos Evaluar restricciones, Ver restricción y Listar restricciones, los restantes se encuentran en el Anexo #4.

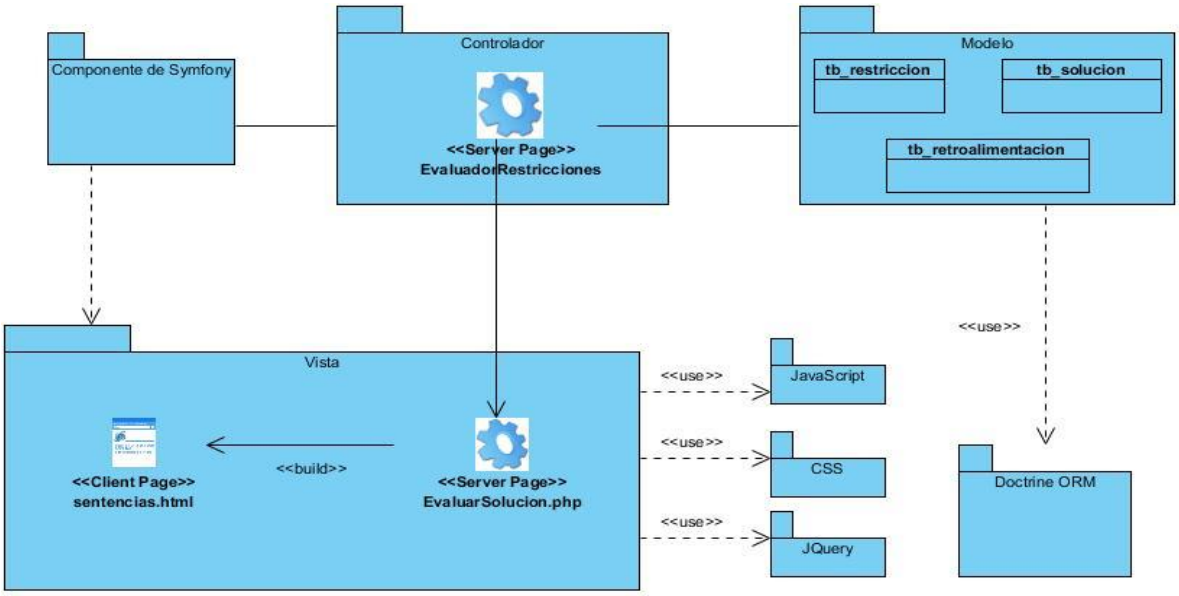

Imagen 20.DCD Evaluar restricciones.

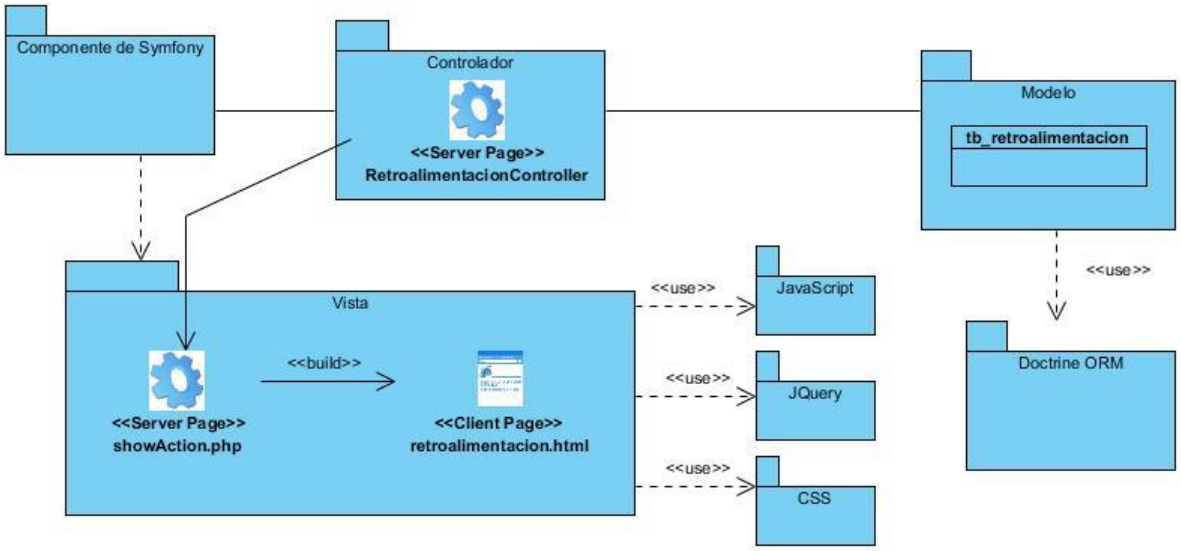

Imagen 21.DCD Ver restricción.

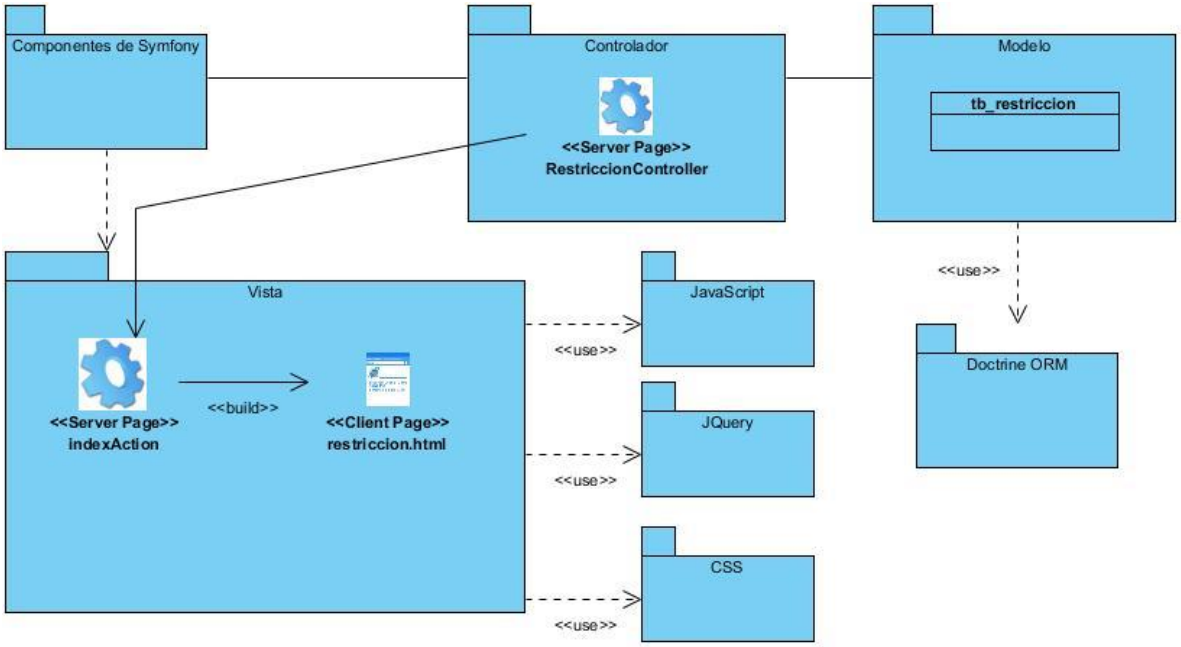

Imagen 22.DCD Listar restricciones.

3.5 Diagrama de secuencia del diseño (DS)

Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal, donde cada rol en la secuencia se muestra como una línea de vida que representa el rol durante cierto plazo del tiempo, con la interacción completa. Los mensajes se muestran como flechas entre las líneas de vida. Un diagrama de secuencia puede mostrar un escenario, es decir una historia individual de una transacción (Jacobson, 2000).

A continuación se muestran los diagramas de Diagramas de secuencia del diseño de los requisitos Evaluar restricciones, Insertar restricción y Listar restricciones, los restantes se encuentran en el Anexo #5.

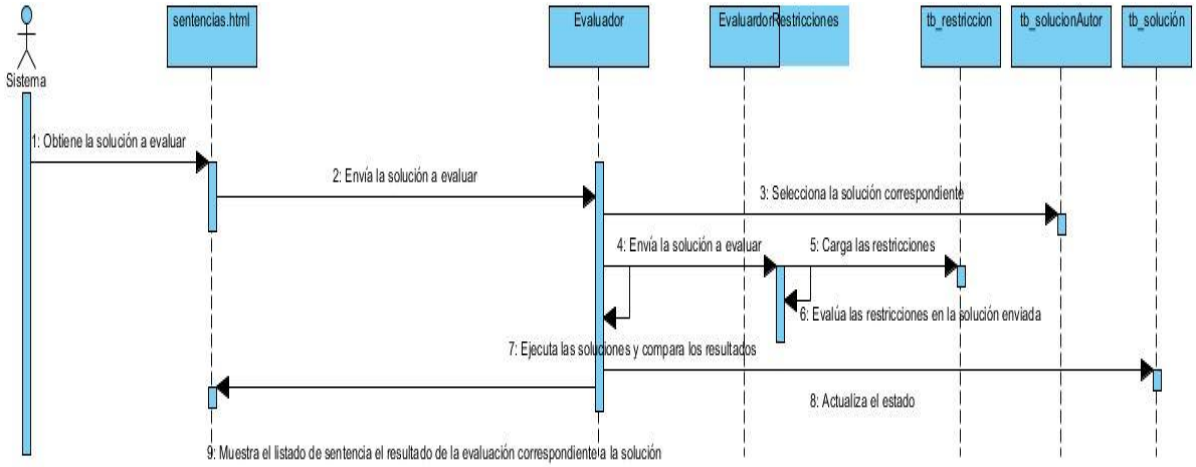

Imagen 23. DS Evaluar restricciones.

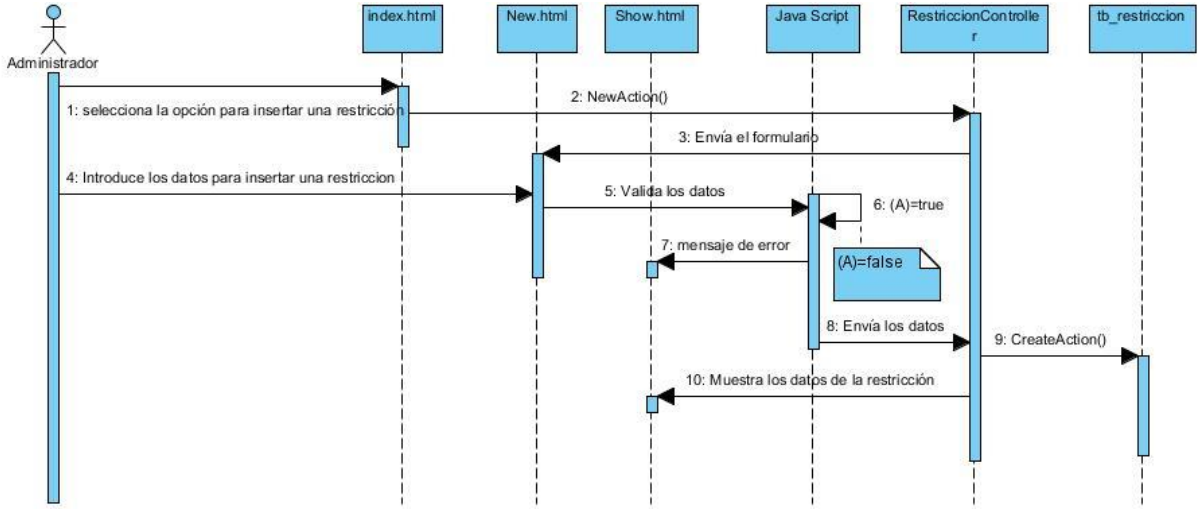

Imagen 24. DS Insertar restricción.

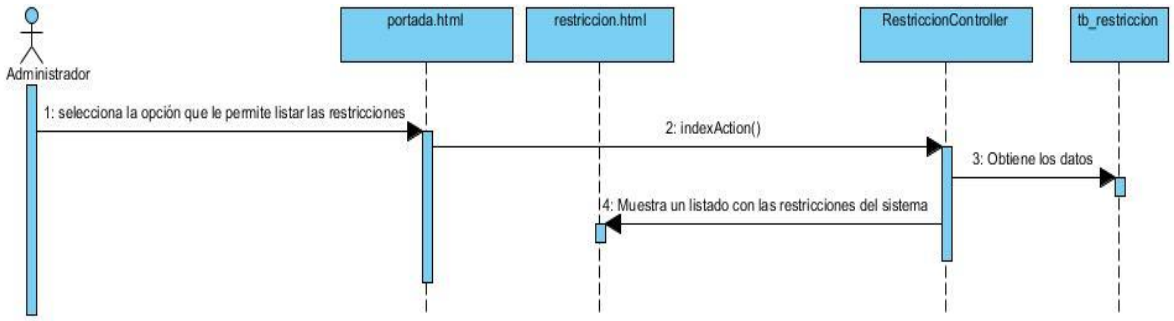

Imagen 25. DS Listar restricciones.

3.6 Modelo de despliegue

El modelo de despliegue define la arquitectura física 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).

3.6.1 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 donde se insertará el componente 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).

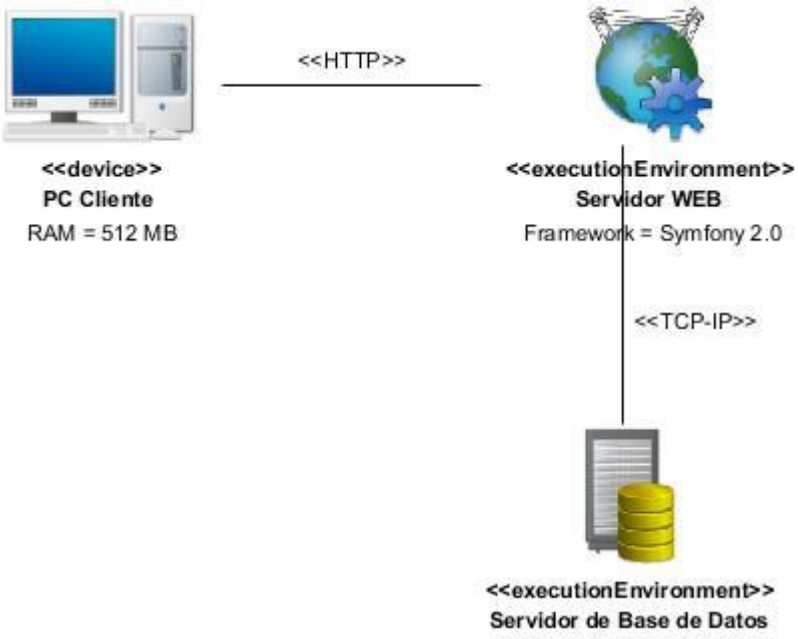

Imagen 26. Diagrama de despliegue del sistema.

3.7 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.

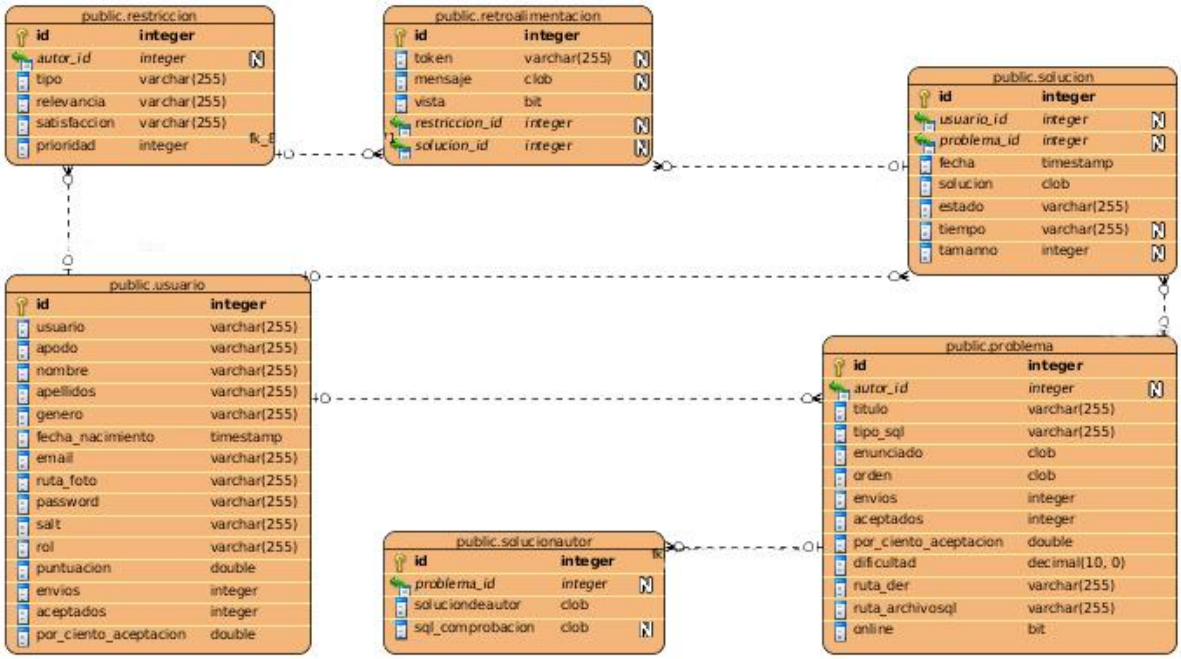

Imagen 27. Diseño de la Base de Datos.

3.7.1 Descripción de las tablas de la base de datos

A continuación se detalla una descripción de la tabla public.restriccion creada para almacenar todos los datos de las restricciones. La descripción de las restantes tablas se muestran en el Anexo #6.

public.restriccion		
Almacena los datos referentes a las restricciones del sistema para evaluar las soluciones de los estudiantes.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Campo que contiene el identificador de la restricción.
autor_id	Integer(10)	Campo que contiene el identificador del usuario que creó la restricción.
tipo	Varchar(255)	Campo que contiene el tipo de sentencia al que se aplicará la restricción.
relevancia	Varchar(255)	Campo que contiene la palabra que representa la condición de relevancia de la restricción.

satisfaccion	Varchar(255)	Campo que contiene la palabra que representa la condición de satisfacción de la restricción.
prioridad	Integer(10)	Campo que contiene un número que representa la dirección en que se evaluará la restricción.

Tabla 5. Descripción de la tabla “restriccion” de la base de datos

3.8 Retroalimentación y evaluación de las soluciones en el 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. 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 Restricciones que analiza todas las soluciones en estado ***Pendiente***. Para dicha evaluación el sistema caga la solución enviada por el estudiante y todas las restricciones asociadas al tipo de problema al que pertenece (SELECT, INSERT, DELETE, UPDATE, VIEW), una vez cargadas, el sistema realiza una evaluación de la solución verificando que no se incumpla alguna restricción que sea aplicable a la solución enviada. Una vez realizado el análisis a la solución, si la misma viola al menos una de las restricciones se califica como ***Error de Compilación*** generándose la retroalimentación asociada a la restricción que desencadenó el error. En caso contrario la solución pasa al estado ***Semántica***.

A continuación se muestra una imagen que representa la retroalimentación que el sistema mostraría si la solución enviada viola alguna de las restricciones:

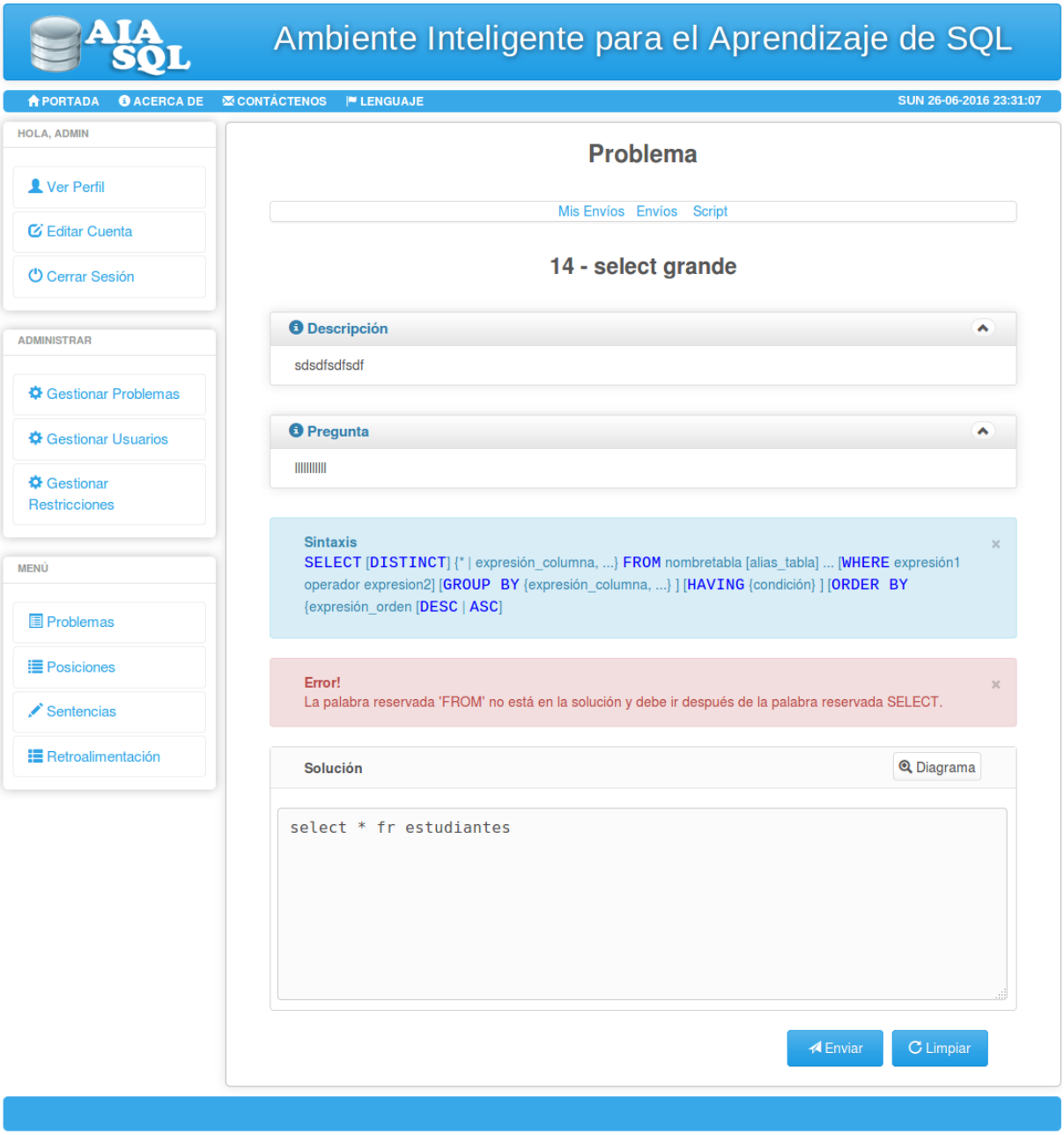

Imagen 28. Datos mostrados en una retroalimentación.

La otra tarea consiste en evaluar el código, se toma 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***.

3.9 Restricciones en el AIA-SQL

Como se explica en el capítulo 1 las restricciones cuentan con un par de condiciones las cuales son Relevancia y Satisfacción, y se explica que la restricción debe ser enunciada de la siguiente manera:

Si la condición de relevancia se cumple la condición de satisfaccion también tiene que cumplirse.

Las restricciones presentes en el sistema actual solo permiten verificar la existencia de tokens en las sentencias enviadas por los estudiantes, un ejemplo de una de estas restricciones es el que se muestra a continuación:

```
private function AnalizarSolucionSelect($tokens)
{
    //Restriccion 7 Sentencia Select
    if (!in_array('select', $tokens)) {
        $this->token = 'SELECT';
        return 'La solución siempre debe comenzar con la palabra reservada SELECT.';

    }

    //Restriccion 7 Sentencia Select
    if (in_array('select', $tokens) && !in_array('from', $tokens)) {
        $this->token = 'FROM';
        return 'Una vez especificados los campos a seleccionar, debes especificar la tabla
            desde donde se van a extraer los datos. Para ello debes utilizar la palabra reservada FROM.';
    }
}
```

Imagen 29. Ejemplo de restricción.

Las restricciones que permite el componente además de verificar la existencia de tokens verifican el orden de los mismos y la existencia de parámetros, así como en caso de aparecer tokens opcionales como GROUP BY y HAVING, el sistema cuenta con la posibilidad de verificar que los mismos estén ubicados correctamente en la solución.

Un ejemplo si el Administrador introduce la siguiente restricción el sistema verificará si aparece el token SELECT también se encuentre el token FROM así como que estén en el orden lineal o sea primero la condición de relevancia y seguidamente la condición de satisfacción, para lograr esto se deben introducir en los campos del formulario los siguientes datos:

Tipo: Select

Relevancia: SELECT

Satisfaccion: FROM

Orden: Lineal

En caso que se desee verificar parámetros se introducirán los siguientes datos:

Tipo: Select

Relevancia: SELECT

Satisfaccion: PARAM

Orden: Lineal

Y la aplicación verificará que después del token SELECT vengan parámetros. En caso que se desee verificar un token opcional se introducirían los datos de la forma que se muestra a continuación

Tipo: Select

Relevancia: JOIN

Satisfaccion: FROM

Orden: Inversa

Y la aplicación verifica que en caso de existir el token JOIN tiene que estar después del token FROM. A continuación la se muestra una imagen con las restricciones insertadas en el sistema hasta el momento donde el campo Orden que muestra el valor 1 o 2 se analiza como 1 para verificación lineal y 2 para verificación inversa:

Imagen 30. Listado de restricciones insertadas en el sistema.

3.10 Diagrama de componente (DCOM)

Los diagramas de componentes permiten modelar la vista de implementación del sistema a partir del cual se construye la aplicación, se representan las dependencias entre los componentes para poder determinar el impacto de un cambio y modela además la asignación de clases y otros elementos del modelo a los componentes (Jacobson, 2000).

A continuación se muestran los diagramas de Diagramas de componentes de los requisitos Evaluar restricciones, Ver retroalimentación y Listar restricciones, los restantes se encuentran en el Anexo #7.

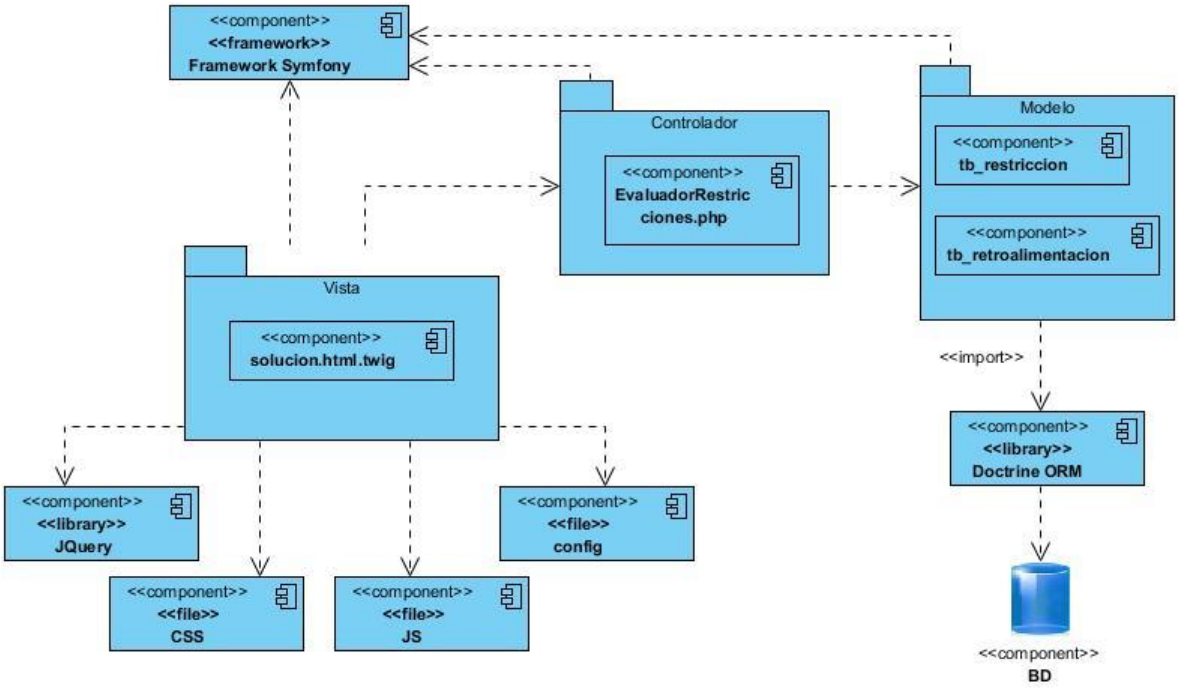

Imagen 31.DCOM Evaluar restricciones.

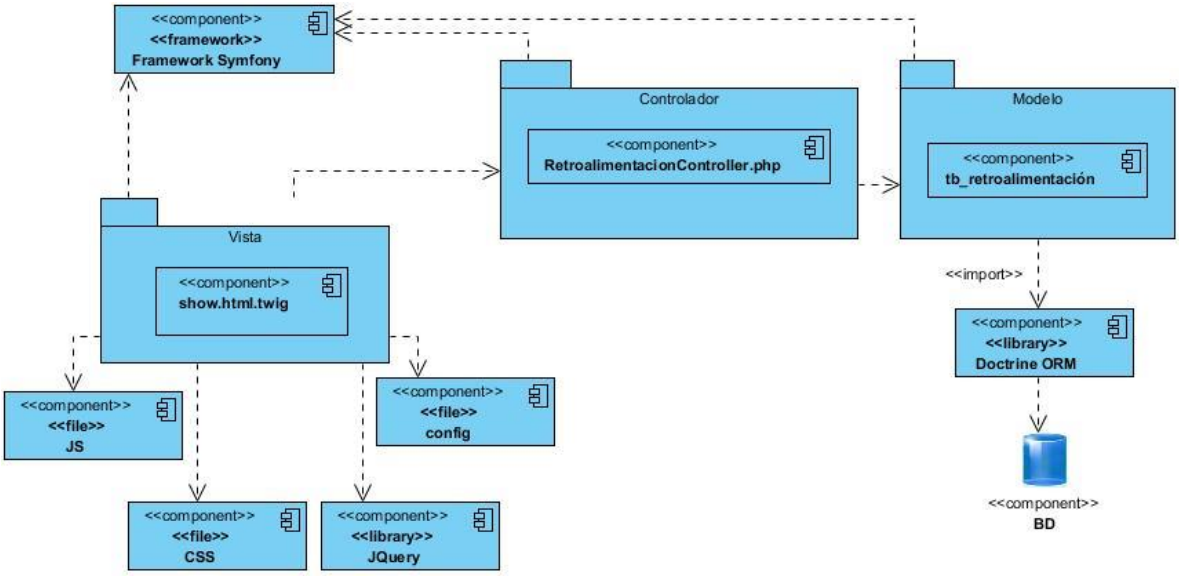

Imagen 32. Ver retroalimentación.

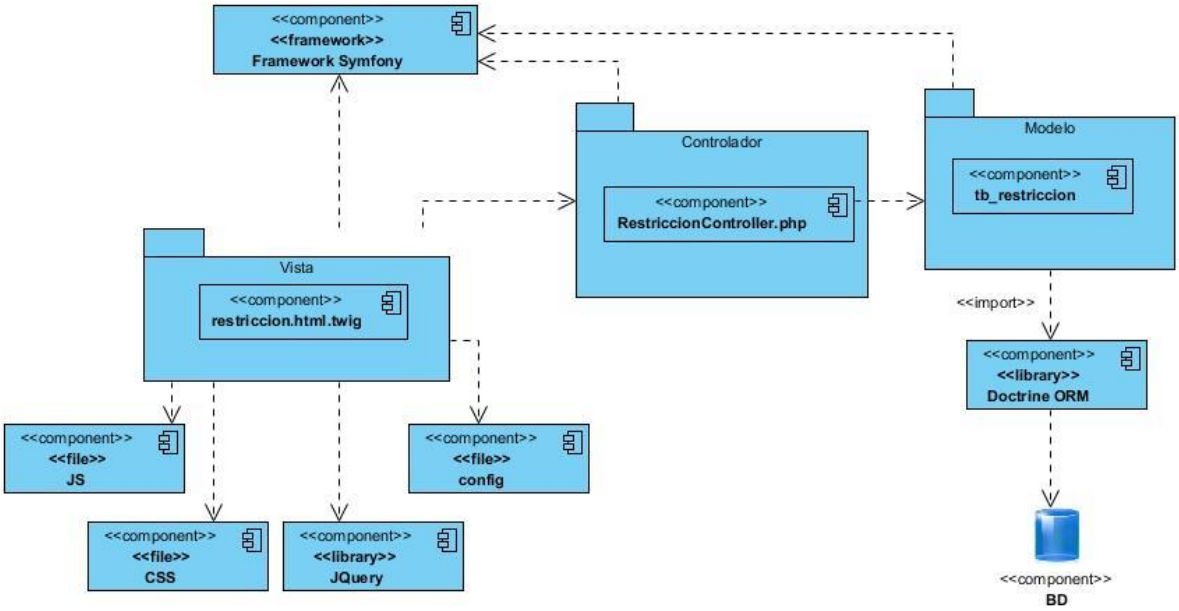

Imagen 33.DCOM de Listar restricciones.

3.11 Estándar de codificación

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Al comenzar un proyecto de software, se debe establecer un estándar de codificación para asegurar de que todos los programadores del proyecto trabajen de forma coordinada y utilizarlo desde el inicio hasta el final del desarrollo (Microsoft, 2016).

Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad es de gran importancia para la calidad del software y para obtener un buen rendimiento (Microsoft, 2016). A continuación se muestran algunos de las pautas definidas para la codificación del componente:

- El nombre de los métodos se definen con la primera palabra en minúscula y todas las palabras siguientes con su primera letra en mayúscula como se muestra a continuación:

```
private function evaluarRestricciones()
{
    switch ($this->tipo_sql) {
        case('select'):
            return $this->analizarSentenciaSELECT($this->tipo_sql, $this->tokens);
            break;
```

Imagen 34. Ejemplo estándar de codificación para el nombre los métodos.

- Los nombres de las variables se definen con la primera palabra en minúscula y todas las palabras siguientes con su primera letra en mayúscula como se muestra a continuación:

```
$tokens = $this->obtenerTokensSQL();
$mensajeFinal = null;
$posInicial = null;
$posFinal = null;
$mensajeSubSentencia2 = null;
```

Imagen 35. Ejemplo estándar de codificación para el nombre las variables.

- Los nombres de las clase se definen con la primera letra de cada palabra en mayúscula como se muestra a continuación:

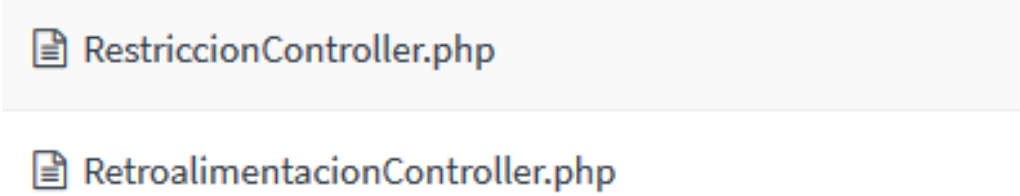

Imagen 36. Ejemplo estándar de codificación para el nombre las clases.

- Las llaves de las clases se deben abrir en la línea siguiente a donde se comenzó a declarar esta, y cerrarse en una línea después del cuerpo de la clase.

```
class EvaluadorRestricciones
{
    private $string_sql, $tipo_sql, $mensaje, $tokens, $sql, $token, $em;

    function __construct($string_sql, $tipo_sql, $em)
    {
```

Imagen 37. Ejemplo estándar de codificación para las llaves en las clases.

- Todos los métodos tienen que estar comentados con la explicación de para qué es usada cada variable que se use.

```
/**
 *$tokens = $this->obtenerTokensSQL(); Guardo en la variable tokens el string de la solución enviada por el
 *estudiante convertida en un arreglo
 *$mensajeFinal = null; Mensaje de error que genera el anÍisis de la solución del estudiante
 *$posInicial = null; Posición inicial de una subsentencia select en la sentencia VIEW
 *$posFinal = null; Posición final de una subsentencia select en la sentencia VIEW
 *$mensajeSubSentencia2 = null; Mensaje que contiene el error en caso de existir en la subsentencia
 * @return null|string
 */
private function analizarSentenciaVIEW()
{
```

Imagen 38. Ejemplo estándar de codificación para los comentarios de los métodos.

3.12 Pruebas

La prueba del software es un elemento de un tema más amplio que suele denominarse verificación y validación. Verificación es el conjunto de actividades que aseguran que el software implemente correctamente una función específica. Validación es un conjunto diferente de actividad es que aseguran que el software construido corresponde con los requisitos del cliente (Pressman, 2005).

Algunos objetivos de las pruebas de software de acuerdo a Pressman son:

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Al realizar las pruebas se comprobará si el sistema cumple con la calidad requerida y los requisitos exigidos por el cliente. Puede que no se obtenga un sistema libre de errores pero si apto para ser usado por el usuario al que es destinado.

3.12.1 Niveles de prueba

El proceso de realización de las pruebas está compuesto por una serie de niveles entre los que se encuentran: el nivel de pruebas unitarias, el de pruebas de integración, el de pruebas del sistema y el de pruebas de aceptación (Pressman, 2002). El componente es sometido a los diferentes tipos de prueba que se describen a continuación.

3.12.1.1 Pruebas de integración

La necesidad de realizar las pruebas de integración viene dada por el hecho de que los módulos que forman un programa suelen fallar cuando trabajan de forma conjunta, aunque previamente se haya demostrado que funcionan correctamente de manera individual. Con el uso de estas pruebas

se consigue formar el programa global a medida que se comprueba como los distintos componentes interaccionan y se comunican libres de errores (Vence, 2010).

3.12.1.2 Métodos de prueba

Existen dos métodos de pruebas que se utilizan con el objetivo de validar el software, método de Caja Blanca y método de Caja Negra, el primero se basa en pruebas que se le realizan al código y el segundo pruebas a la interfaz de usuario. Al componente se le aplicará el método de Caja Negra.

- **Método de Caja Negra**

Las pruebas de Caja Negra son las que se llevan a cabo sobre la interfaz del software. Se concentran en los requisitos funcionales del software. Es decir, permiten derivar un conjunto de condiciones de entrada que ejercitarán los requisitos funcionales de un programa. Encuentran errores en las siguientes categorías: funciones incorrectas o faltantes, errores de interfaz, errores en estructuras de datos o en acceso a bases de datos externas, errores de comportamiento o desempeño y errores de inicialización y término (Pressman, 2002).

Técnicas de prueba

Para desarrollar las pruebas de caja negra se decide escoger la técnica Partición de equivalencia. Basado en esta técnica se elaboran los casos de pruebas que se aplicarán a la aplicación.

Partición de equivalencia

Es una técnica de caja negra que se basa en dividir en subconjuntos equivalentes respecto a una relación específica (clase de equivalencia) el dominio de las entradas. La prueba de un valor representativo de una clase permite suponer que el resultado obtenido será el mismo que para otro valor de la clase. Se realiza un conjunto representativo de casos de prueba para cada clase de equivalencia (Blanco, 2011).

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.

3.12.1.3 Diseño de Casos de pruebas

A continuación se presenta el CP propuesto para el CU Insertar restricción. Para consultar el resto de los CP remitirse al Anexo #8.

CP Insertar restricción
Descripción general: Permitir insertar una restricción en el sistema.

Condiciones de ejecución:

Tener en cuenta los siguientes datos: tipo, condición de satisfacción, condición de relevancia y orden.

Estar autenticado en el sistema con el rol Profesor o Administrador.

Escenario	Descripción	Tipo	Relevancia	Satisfacción	Orden	Respuesta del sistema	Flujo Básico
EC 1.1 Opción Nueva restricción	El usuario selecciona la acción Nueva restricción.	NA	NA	NA	NA	El sistema muestra al usuario los siguientes campos necesarios para insertar una restricción: • (*) Tipo • (*) Relevancia • (*) Satisfacción • (*) Orden	Portada/Gestionar Restricciones/Restricciones/Nueva restricción
							Portada/Gestionar Restricciones/Nueva restricción
EC 1.2 Opción Insertar	El usuario llena los datos y selecciona la opción Insertar.	V	V	V	V	Inserta una nueva restricción y muestra la restricción con los datos: • (*) Tipo • (*) Relevancia • (*) Satisfacción • (*) Orden Además se guarda el siguiente dato en la entidad:	Portada/Gestionar Restricciones/Restricciones/Nueva restricción/Insertar
							Portada/Gestionar Restricciones/Nueva restricción/Insertar

						• Autor	
EC 1.3 Opción Volver a la lista	El usuario selecciona del botón Opción Volver a la lista.	NA	NA	NA	NA	El sistema muestra el listado de restricciones	Portada/Gestionar Restricciones/Restricciones/Nueva restricción/Volver a la lista Portada/Gestionar Restricciones/Nueva restricción/Volver a la lista.
EC 1.4 Campos vacíos	El usuario selecciona la opción Insertar y los campos se encuentran vacíos.	NA	NA	NA	NA	El sistema muestra el siguiente mensaje de error: Campos vacíos	Portada/Gestionar Restricciones/Restricciones/Nueva restricción/Insertar Portada/Gestionar Restricciones/Nueva restricción/Insertar

EC 1.4 Campos incorrectos	El usuario selecciona la opción Insertar y los campos especificados son incorrectos.	NA	I	I	NA	El sistema muestra el siguiente mensaje de error: Los valores introducidos no son tokens permitidos por el sistema o la palabra definida para los parámetros "param".	Portada/Gestionar Restricciones/Restricciones/Nueva restricción/Insertar Portada/Gestionar Restricciones/Nueva restricción/Insertar

Tabla 6. CP Insertar restricción

3.13 Resultados de las pruebas realizadas

3.13.1 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, mediante la técnica de partición por equivalencia. Para el desarrollo de la misma se hace uso de los casos de pruebas generados con el fin de detectar la mayor cantidad de no conformidades posibles. A continuación se muestra una tabla resumen donde se muestran la cantidad de no conformidades detectadas (NC), estas a su vez serán divididas en significativas (S) y no significativas (NS).

Funcionalidades	Iteración 1			Iteración 2			Iteración 3		
	NC	S	NS	NC	S	NS	NC	S	NS
Ver restricción	0	0	0	0	0	0	0	0	0
Editar restricción	2	1	1	1	1	0	0	0	0
Eliminar restricción	2	1	1	1	1	0	0	0	0
Insertar restricción	1	1	0	0	0	0	0	0	0
Evaluar restricciones	5	2	3	3	2	1	1	0	1
Generar retroalimentación	3	1	2	2	2	0	0	0	0

Ver retroalimentación	0	0	0	0	0	0	0	0	0
Listar restricciones	1	1	0	0	0	0	0	0	0
Listar retroalimentaciones generadas	1	1	0	0	0	0	0	0	0
Total	15	8	7	7	6	1	1	0	1

Tabla 7. Resultado de las pruebas funcionales por iteración.

A continuación se muestra un gráfico donde se puntualiza por iteraciones el total de no conformidades identificadas, así como significativas y no significativas:

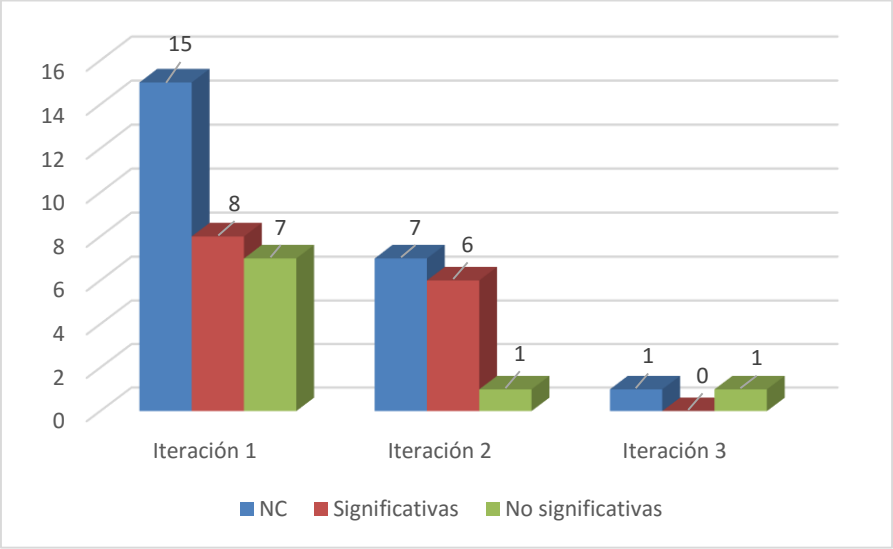

Gráfico 1. Resultado de las pruebas funcionales por iteración.

3.13.2 Resultados de las pruebas de integración

Para la comprobación del funcionamiento del componente desarrollado una vez integrado al sistema, se realizaron dos iteraciones de pruebas de integración. Se detectaron en una primera instancia errores a la hora de realizar la evaluación a las soluciones enviadas por los estudiantes y errores en la retroalimentación generada. La segunda iteración se realizó una vez terminado en su totalidad el componente donde se verificó que se habían solucionado las deficiencias encontradas. En resumen las pruebas de integración validaron el correcto funcionamiento del componente una vez integrado al sistema.

3.14 Conclusiones del capítulo

Con el desarrollo de este capítulo se obtuvieron los siguientes resultados:

- Queda planteando como patrón arquitectónico Modelo-Vista-Controlador que permite la correcta estructura del componente.
- Para la correcta implementación del componente se determinó la utilización de los patrones de diseño: Inyección de dependencias, Iterator, Experto, Creador, Controlador, Decorador, Alta cohesión y Bajo acoplamiento.
- Como artefactos ingenieriles se modelaron los diagramas clases del análisis, de colaboración, de comunicación, de componentes, de secuencia, de despliegue, de clases de diseño y el diseño de la base de datos.
- Con la realización de las pruebas de caja negra mediante la utilización de la técnica participación de equivalencia se detectaron un total de 23 no conformidades en 3 iteraciones las cuales fueron solucionadas correctamente, las resultados pruebas de integración y aceptación resultaron satisfactorias.

Conclusiones

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

- El análisis del estado del arte asociado al objeto de estudio, permitió identificar los conceptos principales para fundamentar teóricamente la propuesta de solución.
- El análisis del Ambiente Inteligente para el Aprendizaje del Lenguaje Estructurado de Consulta permitió identificar la necesidad de implementar un componente que brinde la posibilidad de gestionar de forma automática las restricciones en el sistema.
- El desarrollo del flujo de análisis y diseño correspondientes a la metodología seleccionada, permitió identificar 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, con esto se pudo verificar el correcto funcionamiento de la solución propuesta.
- Se obtuvo un componente que brinda la posibilidad de gestionar de forma automática las restricciones en el sistema y a su vez generar retroalimentación para el apoyo a los estudiantes y profesores.

Recomendaciones

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

- Utilizar la aplicación en competiciones de bases de datos que se desarrollen en la universidad y presentarla en eventos relacionados con el tema.
- Identificar nuevas restricciones e incluirlas en la aplicación.

Bibliografía

Lara. Marilé Lemus Martínez, Yordanis García Leiva,Camilo Fonseca Camejo,Bárbara Almarales Lara Sistema para la Informatización de la Gestión de Anuarios. 2016. 1, La Habana : Ediciones Futuro, 2016, Vol. 9. ISSN: 2306-2495.

Núñez. Adrián Peña-Peñate, Luis Guillermo Silva Rojas, Rubén Alcolea Módulo de filtrado y segmentación de imágenes médicas digitales. 2016. 1, La Habana : Revista Cubana de Ciencias Informáticas, 2016, Vol. 10. ISSN: 2227-1899.

Almaguer. PROFESIONAL, Osmany Aguilera Almaguer IMPACTO DE LA TECNOLOGÍA EDUCATIVA EN LA FORMACIÓN DEL. 2011. 29, s.l. : Cuadernos de Educación y Desarrollo, 2011, Vol. 3.

Alvarez, Miguel Angel. 2001. desarrolloweb.com. [En línea] 2001. [Citado el: 15 de marzo de 2016.] http://www.desarrolloweb.com/articulos/392.php.

Arguero Zapata, Sara Shadira. 2015. Repositorio Digital ESPE (Universidad de las Fuerzas Armadas). [En línea] 2015. [Citado el: 7 de febrero de 2016.] http://repositorio.espe.edu.ec/bitstream/21000/10306/1/T-ESPE-048835.pdf.

Astudillo, Marcello Visconti y Hernán. 2012. www.inf.utfsm.cl. [En línea] 2012. [Citado el: 10 de 3 de 2016.] https://www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf.

Blanco, Carlos. 2011. Ingeniería de Software - Construcción y pruebas. Universidad de Cantabria. [En línea] 2011. [Citado el: 26 de 5 de 2016.] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-ii/materiales/tema1-pruebasSistemasSoftware.pdf..

Bustamante, Jaime Barbeito Díaz de. 2013. Modelo de enseñanza y aprendizaje de Responsabilidad Social Corporativa con metodología on-line. s.l. : Universidad Politécnica de Madrid Escuela Técnica Superior de Ingenieros Industriales, 2013.

Cabrera, Roberth G. Figueroa Camilo J. Solís Armando A. 2008. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. Loja : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación, 2008.

Campo, Gustavo Damián. 2009. Patrones de Diseño, Refactorización y Antipatrones. Ventajas y Desventajas de su Utilización en el Software Orientado a Objetos. Universidad Católica de Salta : s.n., 2009.

Capps, H. 1988. Foundations of intelligent tutoring systems. s.l. : Lawrence Erlbaum Associates., 1988.

Carmona, Juan Garcia. 2012. Solid y GRASP.Buenas prácticas hacia el éxito en el desarrollo de software. 2012.

Carrillo, Pablo Andrés Díaz Paola Johanna Rodríguez. 2015. Universidad del Valle. Biblioteca Digital. [En línea] 14 de arbil de 2015. [Citado el: 7 de febrero de 2016.] http://bibliotecadigital.univalle.edu.co/bitstream/10893/8357/1/CB-0449487.pdf.

Castillo. Inteligente, Enrique Altuna Método para la construcción del modelo de dominio en un Tutor. 2014. Habana : s.n., 2014, Revista Cubana de Ciencias Informáticas, Vol. Vol 8. ISSN: 2227-1899.

Castillo, Enrique José Altuna. 2015. researchgate. [En línea] 2015. [Citado el: 24 de febrero de 2016.] https://www.researchgate.net/profile/Enrique_Altuna/publication/275212925_Recommender_system_based_upon_k-nn_for_uncertainly_conditions_in_an_Intelligent_Tutoring_System/links/55354cae0cf218056e9294e9.pdf. ISSN:0864-4659, ISSN-e:1606-4924.

Castro. Macías López, Laura M.Castro Validación de tiempos de respuesta usando pruebas basadas en propiedades. 2014. Coruña, España : Universidad de Coruña, 2014.

Colorado, David Andrade Aguilar Eva Mora. 2015. Centro de Estudios e Investigaciones para el Desarrollo Docente. [En línea] enero de 2015. [Citado el: 30 de enero de 2016.] cenid.org.mx/ctes_2015/memorias/index.php/ctes/article/download/220/211. ISSN: 2007- 7475.

Dayvis Malfará, Diego Cukerman, Fernando Cócaro, Juan Pablo Cassinelli, Renzo Séttimo. 2006. Testing en eXtreme Programming. 2006.

Desarrollo ágil de software aplicando programación extrema. Alveiro Rosado Gómez, Alexander Quintero Duarte,Cesar Daniel Meneses Guevara. 2014. s.l. : Ingenio, 2014.

Dr. Santiago Almeida Campos, Dr. Juan Pedro Febles Rodríguez y Lic. Odalys Bolaños Ruiz. 1997. scielo. scielo. [En línea] enero de 1997. [Citado el: 20 de enero de 2016.] http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S0864-21411997000100005.

Entonado, Florentino Blázquez. 2001. SOCIEDAD DE LA INFORMACIÓN Y EDUCACIÓN. Mérida : Consejería de Educación, Ciencia y Tecnología, 2001. I.S.B.N.:84-95251-60-4.

FORTES, Centro. 2008. Propuesta de proyecto: Soluciones de Apoyo a la Programación Competitiva. La Habana : s.n., 2008.

Gallego, Antonio Javier. 2016. Introducción a la programación web. 2016.

Goggi, Lucila Ana Cuccaro. 2012. Adecuación de la metodología de desarrollo Extreme Programming a proyectos llevados a cabo en la materia Laboratorio III de la Facultad de Ingeniería de la Universidad Austral. Austral : s.n., 2012.

Guti, Anyel. 2016. Aprende Web. [En línea] 29 de 1 de 2016. [Citado el: 5 de febrero de 2016.] http://aprende-web.net/progra/ajax/ajax_1.php.

Jacobson, Ivar Rumbaugh James Booch Grady. 2000. Lenguaje Unificado de Modelado. Manual de referencia. Madrid, España : Addison Wesley, 2000. ISBN 84-7829+037-0.

Jain, Sanil. 2015. ESC101-ITS . Kanpur : s.n., 2015.

JetBrains. 2016. JetBrains. [En línea] enero de 2016. [Citado el: 7 de febrero de 2016.] https://www.jetbrains.com/phpstorm/help/meet-phpstorm.html.

José Joskowicz. 2008. Reglas y Prácticas en eXtreme Programming. 2008.

Joskowicz, José. 2008. Reglas y Prácticas en eXtreme Programming. 2008.

Junco, MSc. Tomás Orlando. 2012. UN JUEZ EN LÍNEA AJUSTADO A LAS NECESIDADES DE LA DOCENCIA. La Habana : s.n., 2012.

Larman, Craig. 1999. UML and Patterns. 1999.

—. 2003. UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado. s.l. : Pearson Educación, 2003. 9788420534381.

Larman, Craig, Hall, Prentice. 2003. UML y Patrones. 2003.

Letelier, Patricio. 2006. Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP). 2006.

Luis Miguel Echeverry Tobón, Luz Elena Delgado Carmona. 2007. Caso práctico de la metodología XP al desarrollo de software. Pereira : s.n., 2007.

Ma, Rafael. 2010. PostgreSQL-es. [En línea] 2 de 10 de 2010. [Citado el: 30 de enero de 2016.] http://www.postgresql.org.es/sobre_postgresql.

Marco, Jordi. 2013. Implementando acceso directo y secuencial a colecciones de datos mediante aspectos. Catalunya : s.n., 2013.

Masó, Josep Soler. 2010. ENTORNO VIRTUAL PARA EL APRENDIZAJE Y LA EVALUACIÓN AUTOMÁTICA EN BASES DE DATOS. s.l. : Universidad de Girona, 2010. 978-84-694-0260-3.

Mateu. web, Carles Mateu Desarrollo de aplicaciones. 2004. Barcelona : Fundació per a la Universitat Oberta de Catalunya, 2004. ISBN: 84-9788-118-4.

Microsoft. 2016. Microsoft Developer Network. [En línea] 20 de abril de 2016. [Citado el: 15 de junio de 2016.] https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx.

Mitrovic, A. 1998. “Learning SQL with a computerized Tutor”. Atlanta, Georgia : s.n., 1998. DOI: 10.1145/273133.274318.

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

Mitrovic, Antonija. 2012. Fifteen years of constraint-based tutors: what we have achieved and where we are going. 2012. DOI: 11257-011-9105-9.

Moritz, S. 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. Canada : s.n., 2008. ISBN: 847-9901-4777.

Naveda, A.Cortez C. 2011. ESTRATEGIAS PARA EL TRATAMIENTO DE ANTIPATRONES EN LOS MODELOS DE SOFTWARE. Argentina : s.n., 2011.

Patterson, Owen Briggs Steven Champeon Eric Costello Matt. 2003. Cascading Style Sheets. España : ANAYA MULTIMEDIA, 2003. ISBN: 978-84-415-1497-3 84-415-1497-6.

Penadés, Patricio Letelier Carmen. 2006. [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm. ISSN 1666-1680.

Pérez, Javier Eguíluz. 2009. Introducción a JavaScript . 2009.

Polson, M, Richarson, J. 1988. Foundations of Intelligent Tutoring Systems. Hillsdale : Lawrence Erlbaum Associates Publishers, 1988.

Potencier, François Zaninotto Fabien. 2008. Symfony 1.2, la guía definitiva. 2008.

Pressman, Roger. 2002. Ingeniería de Software: Un enfoque práctico 5ta edición. s.l. : McGraw Hill, 2002. ISBN: 8448132149.

—. 2005. Ingeniería del Software. s.l. : MCGRAW-HILL, 2005. ISBN: 9701054733.

Prior, Fernando Flores. 2015. Visor de información de telemetría. México : IMTA. Coordinación de Comunicación, Participación e Información. Subcoordinación de Difusión y Divulgación, 2015.

Pumarejo, Egberto D. Torres Marlyn González Andrea. 2016. SlideShare. Lenguaje Unificado de Modelado UML. [En línea] 2016. [Citado el: 10 de Enero de 2016.] http://es.slideshare.net/AndreaPumarejo/lenguaje-unificado-de-modelado-uml.

Ramirez, Frank Jibaja. 2015. Metodologías para el desarrollo de software. 2015.

Ruiz, Francisco. 2000. Modelo Entidad-Relación. s.l. : UCLM-ESI, 2000.

Sánchez, David Leonardo Rodríguez Bello David Arturo Valero. 2015. porticus.usantotomas.edu.co. [En línea] 2015. [Citado el: 29 de enero de 2016.] http://porticus.usantotomas.edu.co:8080/xmlui/bitstream/handle/11634/391/Adaptacion%20de%20una%20solucion%20de%20software%20libre%20para%20el%20control%20y%20monitoreo%20de%20traslados.pdf?sequence=1&isAllowed=y.

Sánchez, Marta Martín. 2015. Escuela Técnica Superior de Ingenieros de Telecomunicación. [En línea] 2015. [Citado el: 30 de enero de 2016.] http://oa.upm.es/37330/7/PFC_MARTA_MARTIN_SANCHEZ_2015.pdf.

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 Informaticas, 2013.

Sommerville, Ian. 2005. Ingeniería de Software (Parte IV Desarrollo). 2005. 84-7829-074-5.

—. 2005. Ingeniería de Software (Parte IV Desarrollo). 7ma. s.l. : PEARSON Addison-Wesley, 2005. ISBN 84-7829-074-5.

Suárez, Eliezer Cruz. 2015. Desarrollo de una aplicación para visualización de predicciones meteorológicas en dispositivos móviles. La Laguna : Universidad de la Laguna, 2015. 5.441.625-L.

Sweat, Jason E. 2005. Guide to PHP Design Patterns. Canada : s.n., 2005. 0-9735898-2-5.

Valdéz, José Luis Cendejas. 2014. Modelo de Desarrollo de Software Integral y Colaborativo. 2014. ISBN-13: 978-84-16036-63-9.

Vanlehn, Kaleb, Lynch, Charles. Lessons, The Andes Physics Tutoring System:. 2005. 2005.

Vence, Jose. 2010. Plan de pruebas de integración. Madrid, España : s.n., 2010.

Wegner, E. 1987. Artificial intelligence and tutoring systems. Computational and Cognitive Approaches to the Communication of Knowledge. Los Altos, California : Morgan and Kaufman, 1987.