Universidad de las Ciencias Informáticas
Facultad 5

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

Título:

"SIA-MD: Sistema basado en casos para el trabajo didáctico con estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I"

Autores:
Ofelia Peláez Soto
Yeinelis Hierrezuelo Ramírez

Tutores:
MSc. Yidian Castellanos Sabarí
Ing. Raudel Arencibia Ramírez

Consultantes:
Ing. Amado Naranjo Comas
Ing. Liyanis Peláez Baños

La Habana, junio de 2016

Declaración de autoría

Declaramos ser autoras de la presente tesis y reconocemos a la UCI los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de _____ del año 2016.

_____	_____
Firma del autora	Firma del autora
Ofelia Peláez Soto	Yeinelis Hierrezuelo Ramírez
_____	_____
Firma del tutor	Firma del tutor
MSc. Yidian Y. Castellanos Sabarí	Ing. Raudel Arencibia Ramírez

Dedicatoria

A mis hermanos: mi flaco Orlando y mi negro Brian Carlos y mi querida hermana Ayamey, a mi niña Angie, y mi cuñado Michel quiero dedicar el cumplimiento de este proyecto, ya que cada uno de ellos fue mi inspiración para formarme como profesional. Espero que cada uno de ellos alcance esta meta algún día.

Yeinelis Hierezuelo Ramìrez

A mi mami, papi y Elkito.

A mi Rorri.

Ofelia Peláez Soto.

Agradecimientos

Quisiéramos agradecer en conjunto a personas que han contribuido a concluir con éxito un proyecto que en principio parecía una tarea titánica e interminable: al profe Marvyn, Lefebre e Isbel del XETID, a Victor y Eddry. Agradecimiento especial a Antonio que aunque su ausencia se note mucho, llegó para quedarse en nuestras vidas. Para este buen amigo y compañero, deseamos el cumplimiento de todas sus metas y queremos señalar que nunca lo olvidaremos porque siempre tendrá un lugar en nuestros corazones. Un agradecimiento singular le debemos a Liyanis Peláez, Raudel Arencibia y Amado Naranjo que contribuyeron en la realización de este trabajo con sus orientaciones, apoyo y correcciones, por su interés y entrega GRACIAS.

Agradecimientos de Yeinelis

A mi mamá que me ha enseñado que cada experiencia sea agradable o no, es una enseñanza que te forja como persona. A ti mami, muchas gracias. En especial a Yuliesis por ser la primera persona que confió en que yo podía seguir una carrera universitaria. Te quiero mucho Yula. A mi novio Yoelvis por la dedicación, el apoyo, la seguridad y la calma que me brinda ante cada situación que hemos enfrentado en lo últimos 5 años.Te quiero mucho "Chichi".

A toda mi familia por el apoyo y el cariño que me han brindado en cada momento, gracias por entenderme, quererme, y creer en mí. En especial a mi tío Armando y a mi tía Magaly. A mi madrasta Caridad que impulsó a continuar con mi mi carrera a pesar de cada dificultad que se me presento, por el apoyo y poner tus manos en el fuego por mí, Gracias Cary. A mi papá por confiar en que yo puedo alcanzar cualquier proyecto que me proponga, gracias por los consejos y por aceptarme y defenderme como tu hija. A mis suegros Raisa y Carlos al Miche por acogerme como su propia familia muchas gracias.

Agradecimientos de Ofelia

A mami y papi por ser los cimientos de mi formación, ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida. Me formaron con reglas y ciertas libertades, pero al final de cuentas, me motivaron con constancia para alcanzar mis anhelos. Por su incondicional apoyo y amor perfectamente mantenido a través del tiempo, gracias.

Parece como si nunca hubiéramos estado en paz, siempre batallando por cualquier cuestión, sin embargo, llegaron los momentos en los que nuestra lucha cesó e hicimos una tregua para lograr metas conjuntas. Te agradezco no solo por estar presentes aportando

buenas cosas a mi vida, sino por los grandes lotes de felicidad y de diversas emociones que siempre me has causado, muchas gracias mi hermano: Elkito.

La ayuda que me has brindado ha sido sumamente importante, estuviste a mi lado inclusive en los momentos y situaciones más tormentosas, siempre ayudándome. No fue sencillo culminar con éxito este proyecto, sin embargo siempre fuiste muy motivador y esperanzador, me decías que lo lograría perfectamente. Me ayudaste hasta donde te era posible. Muchas gracias por tu amor y apoyo: Raudel.

Cuando me preguntan cuántos hermanos tengo, siempre respondo dos, uno de sangre y uno de corazón. Y tú eres ese segundo hermanos que ha llegado a mi vida como una mano derecha, te agradezco por todas tus ayudas y aportes al igual que todos los buenos momentos pasados. Muchas gracias por ser así, una gran persona, nunca cambies: Daynier.

Tú, amiga has estado a mi lado desde que éramos unas niñas risueñas, El tiempo sigue pasando y ahí estas, cerca ofreciendo lo mejor que tienes, gracias por tu apoyo, por tus esfuerzos y por mantener siempre viva la amistad. Gracias Esme a ti y a tu familia por acogerme como una hija más.

La amistad es uno de los valores más importantes con los que puedas contar, y es que este se manifiesta a través de personas especiales como Yaselis, amigas de innumerables riñas, de confesiones, de apoyo. Simplemente gracias por estar y nunca dejes de reír. Hello kitty te quiero mucho.

Resumen

La presente investigación tributa a la asignatura Matemática Discreta I que se imparte en la Universidad de las Ciencias Informáticas como parte del plan de formación de sus ingenieros. Tras constatar que la experiencia pedagógica de los profesores no era utilizada para beneficiar la toma de decisiones en la orientación de los remediales que se aplicaban y conocer que se omitía el análisis para determinar si un estudiante presentaba deficiencias similares a otros de cursos precedentes se utilizaron algunos métodos científicos para revelar como problema de la investigación: ¿Cómo contribuir a la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I?.

A partir del estudio de los Sistemas basados en casos y los algoritmos de agrupamiento, se desarrolló para solucionar esta problemática un sistema que permite la integración de estas dos ramas de la Inteligencia Artificial. Por tanto, se presenta como objetivo general: desarrollar un Sistema basado en casos, para el apoyo a la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en esta materia. Se obtuvo como resultado un sistema de recomendación que apoya la toma de decisiones por parte de los profesores en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I.

Palabras claves: algoritmos de agrupamiento, inteligencia artificial, matemática discreta, razonamiento basado en casos.

Summary

This research contributes to the subject Discrete Mathematics which is taught in the University of Information Science as part of the training plan of its engineers. Noting that the pedagogical experience of the teachers was not used to benefit decision making in guiding the Remedial that applied and found that the analysis omitted to determine if a student had similar deficiencies to others in precedents courses used some scientific methods to reveal as research problem: How to contribute to decision making in the allocation of remedial students with poor academic performance in the subject Discrete Mathematics I?.

From the study of cases based systems and clustering algorithms, developed to solve this problem a system that allows the integration of these two branches of Artificial Intelligence. Therefore, it is presented as a general goal: to develop a case-based, to support decision making in the allocation of remedial students of low achievement in this area system. This research obtained as a result of a recommendation system that

supports teacher’s decision-making in the allocation of remedial students with poor academic performance in the subject Discrete Mathematics.

Índice

Índice de figuras

Índice de tablas

Índice de ecuaciones

Introducción

El dinamismo de la sociedad actual conlleva a que las personas necesiten constantemente tomar decisiones, por ende, se ha convertido en una tarea fundamental presente en los diversos contextos de la vida personal, familiar y laboral.

La toma de decisiones se concibe como "el proceso para identificar y solucionar un curso de acción para resolver un problema específico" (Stoner, 2003). Sin embargo, esta tarea ante situaciones complejas a menudo supera las capacidades cognitivas del ser humano; donde el juicio intuitivo y su toma de decisiones están lejos de ser óptimo, viéndose deteriorado con la complejidad y la subjetividad humana. Debido a la importancia de la toma de decisiones en muchas situaciones, ofrecer una ayuda que mitigue las deficiencias anteriores constituye una de las líneas principales de la ciencia.

El desarrollo de la informática ha tenido un impacto significativo en esta cuestión por las posibilidades que brinda para analizar, almacenar y manipular grandes cantidades de información. La misma permite la integración con otras áreas de la ciencia como la Inteligencia Artificial[1] para apoyarse en modelos como el Razonamiento basado en casos (RBC), el cual se utiliza en la toma de decisiones a partir de ejemplos de situaciones y decisiones pasadas (Kolodner, 1992).

En la educación se requiere de la toma de decisiones para una efectiva y eficiente formación cognitiva y emotiva-afectiva de las personas que participan en el proceso educacional. En este aspecto se pudiese referir que dicha tarea tiene como fin, la elección entre diferentes alternativas para la solución de diversas situaciones que se presentan en el marco de la gestión de los centros escolares, de forma individual o grupal (Ramírez Blanco, y otros, 2013).

Las instituciones universitarias cubanas, constituyen ejemplos de estos centros de educación, las cuales tienen como misión formar profesionales calificados y desarrollar la investigación científica indistintamente de las diferentes profesiones que ellas comprenden. La Universidad de las Ciencias Informáticas (UCI), tiene como objetivo: "formar ingenieros en ciencias informáticas, que cultiven los valores que caracterizan al ciudadano revolucionario cubano y que alcancen competencias en su campo profesional, sean creativos, honrados, solidarios, y desarrollen un espíritu crítico, auto-crítico, como también de auto-superación" (UCI, 2002). El plan de estudio de esta institución tiene una duración de cinco años y está compuesto por catorce disciplinas

[1] El área multidisciplinaria, que a través de ciencias como las de la computación, la matemática, la lógica y la filosofía, estudia la creación y diseño de sistemas capaces de resolver problemas cotidianos por sí mismos utilizando como paradigma la inteligencia humana (Stuart , y otros, 2004).

académicas, una de estas es la de Matemática, dentro de esta disciplina se encuentra la asignatura Matemática Discreta I (MDI). Esta materia es considerada básica en el proceso de formación de los ingenieros, pues provee a los estudiantes madurez en el razonamiento lógico, habilidades para resolver problemas y los fundamentos matemáticos necesarios para comprender otras asignaturas de la carrera. Por eso, la mayoría de los problemas que se pueden resolver con esta son de naturaleza computacional.

El sistema de evaluación de MDI consta de evaluaciones sistemáticas, seminarios, exámenes parciales y finales; a través de las cuales se mide la adquisición de conocimientos de los estudiantes, detectando así sus principales dificultades. Las evaluaciones son almacenadas en diferentes herramientas, donde los datos pueden ser perdurables o no en el tiempo, entre ellas se encuentran: el registro manual de asistencia y evaluaciones sistemáticas, el Sistemas de Gestión Universitaria (Akademos) y el registro conformado con la herramienta Excel que utilizan algunos profesores.

Por otro lado, estos resultados docentes, muestran el grado de dificultad que representa la materia para su entendimiento por parte de cada estudiante. Se desconoce el instante en que el alumno disminuye la aprehensión del conocimiento, por ende, realizar una atención diferenciada en el momento y con los recursos necesarios, para crear una estimulación y orientación adecuada, resulta una tarea engorrosa. No obstante, se aplican una serie de acciones, (remediales), tales como: consultas, preguntas escritas, tareas personalizadas y encuentros comprobatorios, para lograr disminuir las dificultades que por etapas presentan los estudiantes. Sin embargo, aún persisten dichas dificultades, de las cuales se deriva la **situación problemática** siguiente:

- La experiencia pedagógica de los profesores no es utilizada para beneficiar la toma de decisiones en la orientación de los remediales.
- Se omite el análisis para determinar si un educando presenta deficiencias similares a otros de cursos precedentes, lo que implica que los remediales aplicados sean poco exitosos.

Los elementos anteriormente planteados evidencian la necesidad de mejorar la dinámica del proceso de toma de decisiones en la asignación de remediales a estudiantes de bajo aprovechamiento académico en la asignatura Matemática Discreta I. Por lo que se enuncia como **problema investigativo**:

¿Cómo contribuir a la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I?

Lo cual determina como **objeto de estudio**: aplicaciones informáticas para el apoyo a la toma de decisiones en la educación superior.

Para darle solución al problema planteado se formula como **objetivo general**: desarrollar un Sistema basado en casos, para el apoyo a la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I.

Lo antes expuesto determina como **campo de acción**: los Sistemas basados en casos para la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I.

Para darle cumplimiento al objetivo general se proponen las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre los Sistemas basados en casos.
- Definición de los requisitos del sistema.
- Elaboración de la Base de conocimientos que respalda la información del sistema.
- Representación del conocimiento (expertos) en la Base de conocimientos.
- Determinación de las funcionalidades del sistema.
- Implementación de las funcionalidades propuestas.
- Realización de pruebas al sistema desarrollado.

Para guiar la investigación se utilizan los siguientes métodos científicos:

Métodos teóricos:

- Analítico-Sintético: en el análisis y resumen de documentos donde se expone información referente al Razonamiento basado en casos y su contribución a la toma de decisiones.
- Histórico-Lógico: para realizar un estudio cronológico y explícito de las tendencias actuales sobre los sistemas informáticos que apoyan el control docente.
- Modelación: para representar los prototipos de interfaz de las diferentes vistas en el diseño de la aplicación informática y en modelamiento de la base de conocimiento.

Métodos empíricos:

- Consulta de la información en todo tipo de fuente: en la elaboración del marco teórico de la investigación, que sirve de guía para el resto del trabajo.
- Observación: para comprender la forma en que los profesores de la asignatura Matemática Discreta I realizan el control docente.

- La entrevista: para obtener la información necesaria con la finalidad de identificar las funcionalidades del sistema y para obtener los rasgos predictores y objetivo que componen un caso.
- Pruebas: para verificar la viabilidad de la propuesta de solución obtenida.

Se propone como **idea a defender**: el desarrollo de un Sistema basado en casos contribuirá a la toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I.
Se plantean como **resultados esperados**:

- Elaboración de la Base de conocimientos que sustente el sistema.
- La elaboración de un Sistema basado en casos para el apoyo al proceso de toma de decisiones en la asignación de remediales a estudiantes de bajo rendimiento académico en la asignatura Matemática Discreta I.

El presente trabajo consta con la siguiente estructura capitular:
Capítulo I. “Marco teórico de la investigación”: se realiza una descripción de los módulos del Razonamiento basado en casos y los algoritmos de agrupamiento. Se fundamenta la selección de la metodología y tecnologías de desarrollo de software. Se abordan conceptos fundamentales necesarios para la comprensión de la investigación e incluyen los resultados del estado del arte de las soluciones existentes en la actualidad.
Capítulo II. “Fases planificación y diseño”: se presenta la propuesta de solución para la herramienta y se realizan las fases de planificación y diseño de la metodología de *software*, así como los artefactos generados por cada una de estas fases.
Capítulo III. “Fases desarrollo y prueba”: se presentan las tareas de ingeniería desarrolladas y los resultados de las pruebas unitarias para las funcionalidades conjuntamente a las pruebas de aceptación correspondientes a las historias de usuario.

Capítulo I. Marco teórico de la investigación

Introducción

En el presente capítulo se realiza una caracterización de las distintas soluciones existentes que están en correspondencia con el objeto de estudio de la investigación. Se ilustran las principales ventajas y desventajas del Razonamiento basado en casos para la clasificación. Se describen las características fundamentales de las herramientas y tecnologías de *software* utilizadas para el desarrollo del sistema.

1.1. Análisis de soluciones existentes

En el transcurso de la investigación se estudiaron varios sistemas en el ámbito internacional y nacional con propósitos similares a los enmarcados en el objeto de estudio de la presente investigación. A continuación se caracterizan los más relevantes.

Ágora: es un producto de software estándar para la gestión de centros docentes y academias de todo tipo, desarrollado por la empresa española Kherian Soft. Se adecua a cualquier tipo de centro o formación, ya sea esta de tipo oficial o de carácter libre (academias de enseñanza general, de idiomas, informática, música u oposiciones). Para ello, la aplicación cuenta con varias versiones y múltiples opciones de organización docente para utilizar en cada caso. Aporta soluciones en diversos sentidos, como la gestión centralizada de datos a través de la cual se gestionan todas las bases de datos de alumnos, profesores, aulas y clases. La generación automática de la documentación permite emitir entregas, recibos y facturas, además de proporcionar listados de soporte de todo tipo. Gestiona con facilidad las excepciones que incluye la planificación: alumnos que sólo pueden venir a la mitad de las clases proyectadas de su grupo o que no se adaptan a ningún grupo y toman clases de varios, clases que cambian de horario, profesores que están de baja y han de ser sustituidos. Con el control automatizado de la docencia real, el programa calcula cada día la docencia que, en función de los horarios y las matrículas, teóricamente se habrá impartido, generando los registros correspondientes, permite al usuario verificarlos periódicamente para ajustar, con un simple clic, las incidencias (clase aplazada, no dada, impartida aunque no estaba prevista) (Ferriol Ortiz, y otros, 2009).

Este sistema posee una apariencia compleja y poco amigable debido a la gran cantidad de funcionalidades que presenta y tiene como inconveniente el alto costo de la licencia para su uso, mantenimiento y soporte.

Power Class: es un sistema que busca mejorar la pedagogía utilizada, a través de la interacción de estudiantes, buscando ser el soporte para formar profesionales con un perfil más competitivo. Desarrollado en la Facultad de Ingeniería en Electricidad y

Computación de Guayaquil, Ecuador. El sistema entre otras funcionalidades permite seleccionar de manera aleatoria un estudiante y a la vez permite evaluarlo, realizando alguna pregunta, almacenando en el sistema si contestó o no correctamente; permite evaluaciones continuas de los estudiantes en las diferentes clases que tenga y mantiene un control de la asistencia diaria de los estudiantes a sus clases (Pulla Quezada, y otros, 2011).

Presenta como inconveniente que al ser un software desarrollado en otro país con licencia privativa no es posible acceder a su código fuente para estudiar y comprender su funcionamiento.

AKADEMOS: el Módulo de control forma parte del Sistema de Gestión Universitaria (Akademos) e interactúa con el resto de los módulos que forman parte de éste. Dentro de las nuevas funcionalidades, permite a los estudiantes interactuar directamente con el registro, de manera que puedan consultar la información de sus notas y asistencia. El sistema almacena registros históricos de las evaluaciones y asistencia de los estudiantes que fueron registradas durante un período. El envío de alertas a estudiantes, profesores, secretarias y directivos docentes es otra de las funcionalidades, alertas que serán mostradas a los usuarios del sistema correspondientes cuando entren al mismo o enviadas por el correo electrónico. Permite a las secretarias generar un reporte final de notas para cada estudiante, que es enviado al Módulo de Expediente de Akademos para ser archivado (Collera, 2005).

El sistema realiza un apropiado control docente en todas las asignaturas que se imparten en la UCI, sin embargo, se hace necesario una aplicación más personalizada para la asignatura MDI que contemple la toma de decisiones en el proceso de asignación de remediales, e intente mejorar la adquisición del conocimiento de los estudiantes en dicha materia.

GECMA: identifica la Aplicación para la gestión y apoyo al control del proceso de enseñanza-aprendizaje de la Matemática Discreta I, surgida en el curso 2014-2015, a partir de una tesis de pregrado cuyo objetivo, según su autor, es apoyar el control del Proceso de Enseñanza Aprendizaje (PEA) de la MDI (Gutierrez Rivero, 2015). La misma contempla el proceso de asignación de remediales, donde las funcionalidades logran un nivel detallado de las evaluaciones y la asistencia de los estudiantes. Posibilita exportar un registro con las evaluaciones de un grupo y la asistencia en las extensiones PDF y DOC. Permite visualizar en un gráfico los porcientos de asistencia e inasistencias de un estudiante. Genera una propuesta de corte evaluativo para cada estudiante.

Constituye una limitante la ausencia de un mecanismo de aprendizaje que posibilite disminuir el tiempo en el proceso de toma de decisiones para la asignación de remediales, mejorar los existentes y reducir los costos asociados a la repetición de

errores. Fue desarrollada con poca flexibilidad: aspectos como el Modelo de planificación y control del proceso docente (P1) no son modificables, de igual forma el profesor no puede agregar un nuevo tipo de evaluación ya que están predefinidas. Los parámetros y los datos almacenados en la aplicación no aportan un significado sustancial a la propuesta de solución.

Las soluciones existentes estudiadas no resuelven el problema de investigación pues presentan como inconveniente que a pesar de realizar una gestión efectiva en los centros docentes donde están desplegados, no cuentan con una base de conocimiento que refleje el trabajo didáctico con estudiantes de bajo aprovechamiento en las asignaturas, no proponen tareas personalizadas para estos estudiantes, ni hacen uso del RBC. Por otra parte, algunos de ellos tienen licencias de carácter privativo por lo que es necesario pagar para hacer uso de los mismos y poder acceder a su código fuente para su estudio y comprensión.

1.2. Razonamiento basado en casos

El RBC, es el método de solución de problemas de los Sistemas basados en casos (SBC). Representa un nuevo método para resolver problemas no estructurados, en el cual el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos (Gálvez Lio, 1998).

La idea básica del RBC es recuperar, adaptar y validar las soluciones encontradas en experiencias previas en un intento de relacionarlas con un problema actual. Las experiencias previas están representadas como una biblioteca de casos que reside en memoria. Cuando se enfrenta con un nuevo problema, el sistema con RBC recupera un caso similar, y la solución del caso se adapta al nuevo problema en un intento para resolverlo (Gálvez Lio, 1998). Posteriormente se realiza un análisis crítico evaluativo en el que se revisa la solución resultante de la adaptación. Por último se realiza el proceso de recordar, este consiste en almacenar el caso en la Base de casos (BC).

Lo anteriormente expuesto es considerado el ciclo de Razonamiento basado en casos como se muestra en la figura 1.

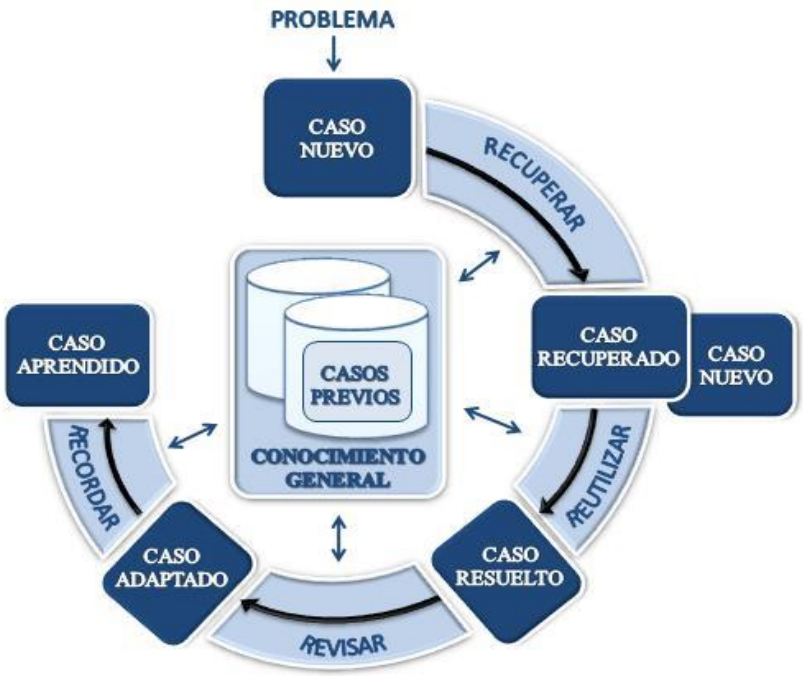

Figura 1: Ciclo del Razonamiento basado en casos. Fuente: Cordero Morales.

1.2.1. Módulo recuperador de casos

Un RBC es bueno en la medida de los casos que pueda recordar. Por eso la recuperación y selección de los casos apropiados constituye un proceso importante, este se pueden realizar mediante dos métodos (Gálvez Lio, 1998):

1. Recuperación por semejanza parcial: este proceso emplea una función de semejanza, que determina una medida numérica del grado de similaridad que tiene un caso con respecto al problema a resolver, esta función considera las diferencias y semejanzas entre la descripción de ambos problemas (la del problema resuelto y la del nuevo problema).

2. Recuperación por analogía: esta busca un caso de la base, cuya descripción se pueda hacer igual a la del problema actual aplicando un reemplazamiento de los valores de los rasgos que tienen valores diferentes, a partir de la correspondencia entre valores definida en una *red semántica*[2].

1.2.2. Módulo de adaptación

Existen métodos generales que son válidos independientemente del dominio de aplicación, y otros que son dependientes del dominio, entre ellos se encuentran (Kolodner, y otros):

[2] Una red semántica o esquema de representación en red, es una forma de representación de conocimiento en la que los conceptos y sus interrelaciones se representan mediante un grafo (Stuart , y otros, 2004).

Métodos de sustitución: se sustituye algunos valores o elementos de la solución antigua por otros adecuados al problema nuevo. Estos métodos van desde la reinstanciación o el ajuste de parámetros hasta la sustitución basada en casos.

Métodos de transformación: se realizan modificaciones más importantes en la solución antigua. Se utiliza reparación guiada por modelos o heurísticas de sentido común que añaden, eliminan o reemplazan algunos componentes de la solución.

Adaptación y reparación de propósito especial: son métodos específicos del dominio que realizan sustituciones, transformaciones estructurales o reparaciones que intentan corregir los fallos que se producen al ejecutar la solución.

Repetición de la derivación: repetir los pasos que se han usado al derivar la solución antigua para obtener la solución al problema nuevo.

Adaptación nula: es la técnica más simple, no es ni estructural ni derivacional. Consiste en no hacer nada y simplemente aplicar cualquiera que sea la solución recuperada, a la nueva situación. Esta es útil en tareas donde el razonamiento necesario para una aplicación puede ser muy complejo, y la solución en sí misma es muy simple.

1.2.3. Módulo de evaluación de soluciones

Los SBC aprenden nuevas experiencias, nuevos casos que aporten un enriquecimiento al sistema. Luego de reutilizar una posible solución esta puede ser correcta o no, si es correcta la nueva solución será almacenada en el sistema, pero si la solución no ha sido satisfactoria entonces se tendrá que revisar. La revisión de casos se realiza en dos fases:

Evaluar la solución: decidir si la solución dada es la correcta al problema planteado. Esta fase normalmente se realiza por un método externo al sistema, por ejemplo, el experto humano, que es una persona competente en un área determinada del conocimiento y cuenta con un largo periodo de preparación y práctica, la cual al aprovecharse se obtienen mejores resultados en el desempeño en cuanto a tiempo, eficacia y precisión en comparación con otra persona no especializada (Ayala, 2006).

Reparar los fallos: si no es correcta la solución se deben detectar los fallos y corregirlos.

1.2.4. Módulo de almacenamiento

Una de las principales características de un sistema con RBC es poder recordar los nuevos casos y su solución aplicada, para ello es fundamental poder retener estos casos en la BC. El primer problema que se debe tratar es decidir qué casos se aprenden. La eficiencia se puede degradar cuando el número de casos crece excesivamente, por tanto, se debe evitar incluir casos que no aporten información nueva al sistema. El rango

de posibilidades va desde los sistemas que, de forma autónoma deciden qué casos deben incluir hasta los que delegan esta posibilidad en el mismo usuario (Arjona, 2006). El segundo problema relacionado con el aprendizaje es el que se refiere a la organización de la estructura de la base de casos. Dependiendo de la complejidad de la estructura utilizada, este proceso puede ser más o menos complicado. Existen diversos enfoques para organizar y modelar la BC, dentro de ellos los más utilizados son los siguientes (Gálvez Lio, 1998):

Organización de los casos en una estructura plana: en este modelo los casos son unidades de información que se organizan en forma secuencial en una lista, arreglo o fichero. Tiene la ventaja de ser poco costoso al añadir casos a la memoria, pero todos los casos son comparados durante la recuperación, por tanto, tienen la desventaja de que este proceso sea costoso. Por eso, este enfoque se utiliza solamente en BC pequeñas.

Organización de los casos en una estructura jerárquica compartida: en este modelo los casos están ubicados en nodos de un árbol. La estructura subdivide los casos de acuerdo a los atributos que comparten. Esto se realiza situando atributos comunes en los nodos internos del árbol. Todos los casos que compartan valores se sitúan debajo de dicho nodo.

Organización de los casos en redes de discriminación: cada nodo es una pregunta que subdivide el conjunto de casos y cada subnodo es una respuesta distinta a la pregunta, esta puede traer otra pegunta para más casos asociados a él. Las preguntas más importantes se colocan en la parte superior. Tiene la ventaja de mejorar la eficiencia en el acceso, pero añadir casos requiere modificar la red, lo que es muy costoso computacionalmente, además algunos casos pueden no ser considerados dado el orden de las preguntas.

1.3. Sistemas basados en casos

Los SBC, son un tipo de *Sistemas basado en conocimiento*[3] cuya forma de representación del conocimiento es mediante casos y el método de solución del problema empleado es el RBC. Existen dos tipos de ellos: los sistemas interpretativos y sistemas resolvedores de problemas, a continuación se profundiza en las características propias de cada uno.

[3] Un sistema computarizado que usa conocimiento sobre un dominio para arribar a una solución de un problema de ese dominio. Esta solución es esencialmente la misma que la obtenida por una persona experimentada en el dominio del problema cuando se enfrenta al mismo problema (Gálvez Lio, 1998).

1. **Sistemas interpretativos:** la interpretación se usa cuando el problema no está bien comprendido. Estos sistemas toman una situación o solución como entrada y su salida es un argumento fundamentando la solución. Son útiles para sistemas de clasificación, evaluación de una solución, argumentación, justificación de una solución, interpretación y predicción de los efectos de una decisión o plan.
2. **Sistemas resolvedores de problemas:** un sistema de este tipo es aplicable cuando una simple presentación de casos resueltos con anterioridad no es suficiente para encontrar una solución; se necesita modificar el caso (la solución dada al caso) para adecuarlo al nuevo problema. Este es el modelo de RBC más empleado para resolver tareas de diseño y planificación (Gálvez Lio, 1998).

1.3.1. Arquitectura de los sistemas basados en casos

Un SBC tiene dos componentes principales: la BC y el resolvedor de problemas.
La BC, contiene las descripciones de los problemas resueltos o no previamente. Cada caso puede describir un episodio particular o una generalización de un conjunto de episodios relacionados. Un caso contiene información útil en un contexto concreto, es una experiencia que enseña. La información de un caso es relativa a determinados objetivos, por lo tanto, se ha de tener en cuenta que una determinada solución puede fallar. En el RBC interesa no solo guardar las soluciones que funcionan sino también aquellas que fallaron, ya que ambas contienen información útil que permitirá repetir las soluciones exitosas, y evitar la repetición de las fallidas (Giménez Arjona, 2006). Generalmente un caso se compone de:

- **La descripción de un problema:** la situación a interpretar a través de un conjunto de variables o rasgos predictores, siendo esta la parte del caso que se emplea para determinar similitudes.
- **La descripción de la solución:** también llamado rasgo objetivo. En esta se puede guardar información adicional en pos de futuras adaptaciones.

El resolvedor tiene a su vez dos componentes: un recuperador de casos y un razonador; la principal diferencia entre la arquitectura de los dos tipos de sistemas con RBC radica precisamente en este último componente. En el tipo solucionador de problemas, el proceso que se ejecuta es "recordar un caso y adaptar su solución" y en el interpretativo el proceso es "recordar un caso y evaluar el problema nuevo basado en su solución"; por lo que en dependencia del tipo el razonador sobre el problema contendrá un fuerte algoritmo de adaptación o un procedimiento de justificación

(Gálvez Lio, 1998).En la siguiente imagen se ilustran los componentes de un SBC:

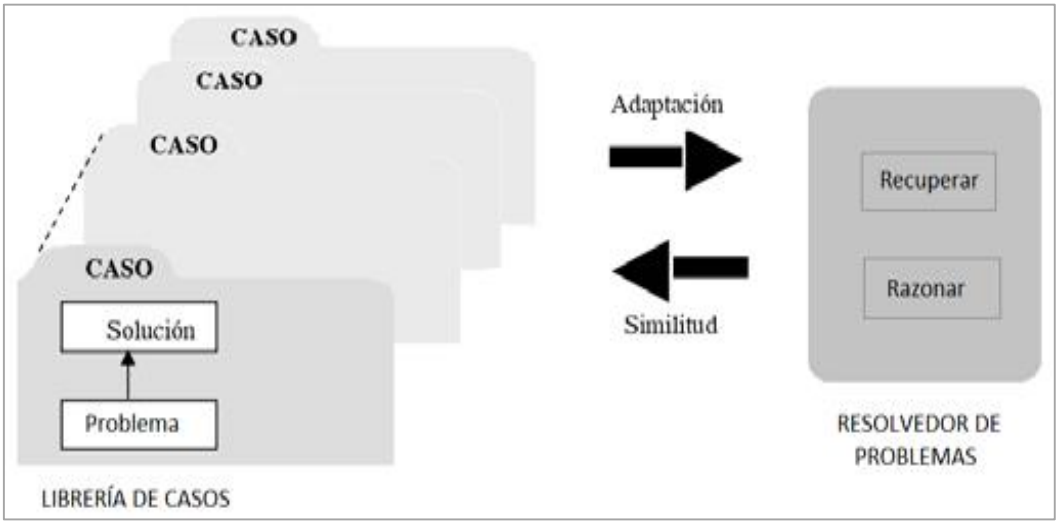

Figura 2: Componentes de un Sistema basado en casos. Fuente: Alberto Ochoa.

1.4. Algoritmos de clasificación

La organización de objetos en categorías es una parte vital de la representación del conocimiento. Aunque la interacción con el mundo tiene lugar a nivel de objetos individuales, la mayoría del proceso de razonamiento tiene lugar en el nivel de categorías (Russell, y otros). Por ello dentro del *aprendizaje automático*[4], se encuentran técnicas de clasificación que permiten agrupar muestras de acuerdo a criterios o métodos, estas técnicas son la clasificación supervisada y la no supervisada.

El objetivo de la clasificación consiste en la asignación de un objeto o un fenómeno físico a una de las diversas categorías o clases especificadas. Se entiende por clase o categoría a una agrupación de objetos que tiene características comunes.

1.4.1. Clasificación supervisada

Este tipo de clasificación cuenta con un conocimiento a priori, es decir para la tarea de clasificar un objeto dentro de una categoría o clase. Se cuenta con modelos ya clasificados (objetos agrupados que tienen características comunes). Se puede diferenciar dos fases dentro de este tipo de clasificación:

La primera fase se tiene un conjunto de entrenamiento o de aprendizaje (para el diseño del clasificador) y otro llamado de *test* o de validación (para clasificación), estos servirán para construir un modelo o regla general para la clasificación.

[4] El aprendizaje automático es una rama de la Inteligencia Artificial que abarca diferentes técnicas que permiten dotar a los computadores de la capacidad de "aprender" modelos computacionales tales que, de forma automática, les permitan resolver nuevos problemas o mejorar su comportamiento en problemas ya vistos (Russell, y otros).

La segunda fase es el proceso en sí de clasificar los objetos o muestras de las que se desconoce la clase a las que pertenecen (Pintos, y otros, 2001).

1.4.2. Clasificación no supervisada

La clasificación es no supervisada cuando se dispone de un conjunto de objetos (observaciones), donde se desconoce tanto el número de clases en que es razonable particionarlo, así como a qué clase pertenece cada observación (Pintos, y otros, 2001). A esta clasificación se la suele llamar también *clustering*[5].

Se emplea el término *clustering* para referirse al análisis de agrupamiento, la tarea de agrupar un conjunto de objetos de tal manera que los objetos en el mismo grupo (llamado un *clúster*) son más similares entre sí que con los de otros grupos.

En este tipo de clasificación se cuenta con objetos o muestras que tiene un conjunto de características, de las que no se conoce a qué clase o categoría pertenece, entonces la finalidad es el descubrimiento de grupos de objetos cuyas características afines permitan separar las diferentes clases (Araujo, 2006).

Entre los algoritmos de agrupamiento se pueden citar los siguientes:

K-means: este algoritmo toma un parámetro de entrada *k*, y parte un conjunto de *n* objetos en *k* grupos, tal que la similitud resultante dentro de un grupo es alta, pero la similitud con otros grupos es baja. Busca una partición óptima de los datos minimizando el criterio de la suma del error cuadrático con un procedimiento iterativo de optimización, el cual pertenece a la categoría de algoritmos *hill-climbing*[6] *o* escalador de colinas.

Ventajas:

- Complejidad computacional lineal O(n*k*I*d), donde **n** es el número de datos, **k** el número de clústeres, **I** el número de iteraciones y **d** el número de atributos.
- Adecuado en grupos compactos y bien separados.

Desventajas:

- Inadecuado para descubrir grupos no convexos, de tamaño y densidad diferente.
- Necesidad de conocer a priori el número k.
- Mínimos locales: dependiente a los centroides iniciales.
- Sensible a la existencia de valores atípicos (Oralla, 2004).

DBSCAN[7] : es un algoritmo basado en métodos de densidad espacial, debido a que define un número de grupos comenzando por una estimación de la distribución de

[5] Del inglés *clúster* que significa grupo en español.
[6] En ciencia de la computación, hill climbing es una técnica de optimización matemática que pertenece a la familia de los algoritmos de búsqueda local.
[7] Density-based spatial clustering of applications with noise

densidad de los nodos correspondientes. Estos grupos están separados por regiones de baja densidad de objetos (ruido).

Ventajas:

- No necesita de la especificación del número de clústeres deseado como lo requiere k-means.
- Puede encontrar clústeres con formas geométricas arbitrarias. Puede incluso hallar un clúster completamente rodeado (pero no conectado) de otro clúster distinto.
- Tiene noción del ruido, y es robusto detectando valores atípicos.
- No es susceptible al orden en que se encuentren los puntos dentro de la base de datos.

Desventajas:

- Asume densidades similares en todos los clústeres.
- Puede tener problemas al separar clústeres pues no puede agrupar conjuntos de datos bien con grandes diferencias en las densidades (Montes, 2013).

Affinity Propagation: este algoritmo determina los clústeres mediante el envío de mensajes entre pares de ejemplos hasta converger. De esta forma un conjunto de datos es descrito empleando un número pequeño de ejemplos, que son identificados como los más representativos del resto de los ejemplos (WANG, 2007).

Ventajas:

- No es necesario especificar el número de clúster a generar.
- Identificación de valores atípicos.

Desventajas:

- La principal limitación de la propagación de afinidad es su exigencia de un espacio de memoria de gran tamaño. El método requiere cuatro de N x N matrices, donde N se refiere al número de elementos a agruparse. Por lo que se recomienda que el número de elementos que deben ser agrupados sea inferior a 1000.
- Requiere un tiempo de O (N^2T) para actualizar los mensajes, donde N y T son el número de objetos y el número de iteraciones, respectivamente. Por lo tanto, se requiere de mucho tiempo de Unidad Central de Procesamiento (CPU), que puede ser excesivo cuando el número de objetos es elevado.

1.5. Metodología de desarrollo de software

Las metodologías de desarrollo de *software* abarcan todo el ciclo de vida del mismo, siendo el "conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo *software*" (Patón, 2006-2007). Debido a la responsabilidad que tiene sobre la vida del *software* es necesario que sea elegido el enfoque adecuado para el desarrollo eficiente de este.

Existen dos enfoques: los ágiles y los tradicionales o prescriptivos. Para la selección del enfoque se realiza el método Boehm y Turner conocido también como el método de la estrella. Este caracteriza al proyecto a partir de 5 criterios, cada uno de estos tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar cuan ágil o prescriptivo debe ser el enfoque a utilizar. Tiene 5 ejes, en cada uno se coloca un criterio, estos se explican a continuación (Serie Científica de la Universidad de las Ciencias Informáticas:Aplicando el método de Boehm y Turner).

Tamaño: este criterio se utiliza para representar el número de personas involucradas en el proyecto. Pueden tenerse en cuenta el nivel de complejidad que pueda presentarse en la comunicación entre los miembros del proyecto y los costos que pueden provocar cambios esperados.

Criticidad: se utiliza para evaluar la naturaleza del daño ocasionado por defectos que no hayan sido detectados al producto. Su evaluación puede ser cualitativa.

Dinamismo: representa la rapidez con la que pueden estar cambiando los requerimientos del proyecto.

Personal: representa la proporción del personal con experiencia alta, media y baja. Los métodos orientados al plan no se ven afectados negativamente por este factor pues no interesa el nivel de experiencia con la que cuenten los miembros del equipo.

Cultura: las organizaciones y las personas que relaciona el proyecto pueden depender de la confianza o de la relación contractual. Esto refleja el nivel de ceremonia necesario y aceptado: documentación, control, formalismo en las comunicaciones.

A continuación se describen el comportamiento de estos criterios en el proyecto:

Tamaño: el equipo de desarrollo está formado por 2 estudiantes de quinto año para la implementación de un total de 24 funcionalidades con ciertos elementos de complejidad, características que permiten clasificar el equipo de desarrollo y al sistema como pequeños. Debido a esta descripción será posible ubicar el punto cercano al centro.

Criticidad: el equipo de desarrollo tiene una elevada responsabilidad con la calidad del producto a obtener, debido al impacto social del mismo. Este sistema permitirá a los profesores de la asignatura MDI asignar remediales, en correspondencias con las características del estudiante. Los errores que se detecten, una vez obtenidos el producto, si bien no provocarán pérdidas de vidas humanas o bienes materiales,

repercutirá negativamente en la utilidad esperada. Por ello el valor de este criterio dentro de la etiqueta se sitúa próximo al centro.

Dinamismo: pueden aparecer cambios en los requerimientos del sistema en cualquier fase del proceso de desarrollo, esto es un riesgo que se asumirá durante todo el ciclo de desarrollo del software. Para mitigarlo, deben adoptarse mecanismos que faciliten la asimilación y adaptación rápida a dichos cambios. Tomando en cuenta esta necesidad como una de las ventajas que brinda el enfoque ágil, se ha ubicado este punto bien cercano al centro de la estrella.

Personal: el valor para este criterio dentro de la estrella se sitúa en 40 % junior porque los desarrolladores no son programadores muy experimentados en las tecnologías a emplear.

Cultura: se posee un buen ambiente entre el equipo de desarrollo, existe buena comunicación y confianza entre sus miembros puesto que han trabajado juntos en otras ocasiones y ambos han cursado 4 años de la carrera en el mismo grupo. Las decisiones tomadas son previamente analizadas y consultadas en conjunto. El equipo presenta plena libertad en el desarrollo del proyecto, así como en la toma de decisiones de este, por lo que se distingue el valor de este criterio como pequeño, muy alejado al enfoque formal o prescriptivo.

La figura 3 muestra la representación de los criterios en la Estrella de Boehm y Turner:

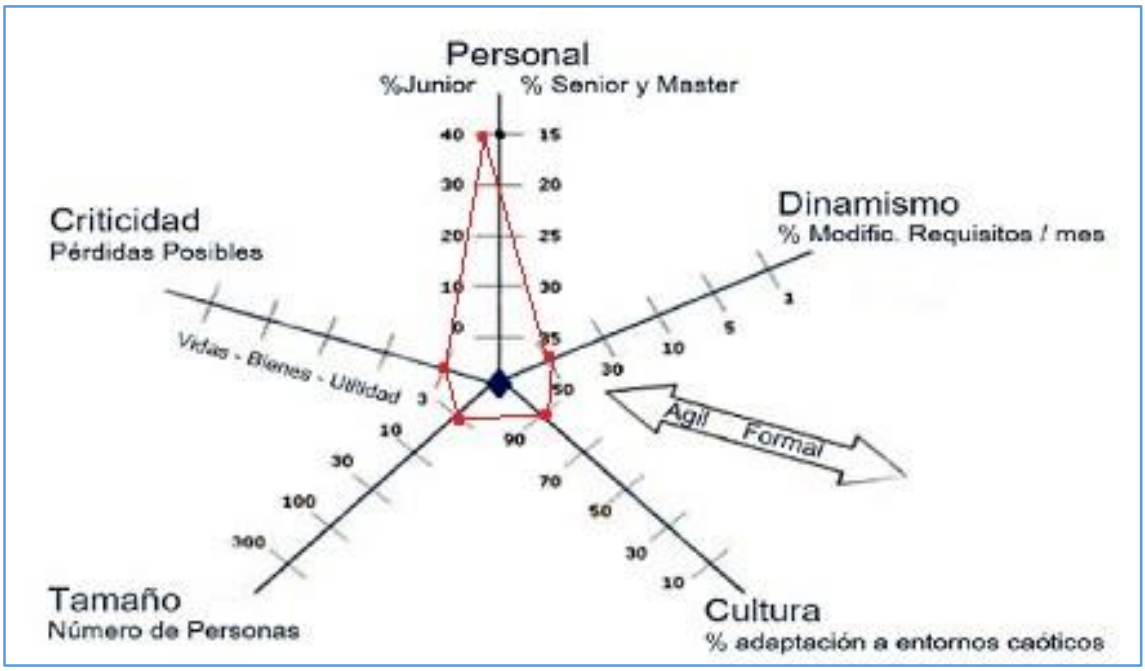

Figura 3: Estrella de Boehm y Turner del proyecto. Fuente: Elaboración propia.

Como se puede apreciar en la disposición de las aristas, se propone adoptar un enfoque ágil. Actualmente existen numerosas propuestas de metodologías ágiles. En la tabla 1,

se observa una comparación entre las metodologías ágiles. Se evidencia una significativa diferencia del índice de agilidad y entre ellas se destaca la Programación extrema, XP por sus siglas en inglés, dentro de las más ágiles adecuándose al desarrollo del Sistema basado en casos.

	CMM	ASD	Crystal	DSDM	FDD	LD	Scrum	XP
Sistema como algo cambiante	1	5	4	3	3	4	5	5
Colaboración	2	5	5	4	4	4	5	5
Características metodológicas (CM):								
- Resultados	2	5	5	4	4	4	5	5
- Simplicidad	1	4	4	3	5	3	5	5
- Adaptabilidad	2	5	5	3	3	4	4	3
- Excelencia técnica	4	3	3	4	4	4	3	4
- Prácticas de colaboración	2	5	5	4	3	3	4	5
Media CM	2.2	4.4	4.4	3.6	3.8	3.6	4.2	4.4
Media Total	1.7	4.8	4.5	3.6	3.6	3.9	4.7	4.8

Tabla 1: Ranking de “agilidad” (Los valores más altos representan una mayor agilidad).

1.5.1. Programación Extrema

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de *software*. Promueve el trabajo en equipo preocupándose por el aprendizaje de los desarrolladores, propiciando un buen clima de trabajo. Se basa en retroalimentación 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. Se considera adecuado su empleo en proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (Beck, 1999).

Al seleccionarse XP como metodología, el proceso de creación del Sistema se verá guiado por las siguientes fases (Solís, 2003):

Planificación: XP plantea la planificación como un permanente diálogo entre las partes la empresarial (deseable) y la técnica (posible). Los clientes plantean a grandes rasgos las Historias de Usuario (HU) que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto (Joskowicz, 2008).

Los artefactos que se generan en esta fase son:

- Historias de usuarios (HU): estas sustituyen a los documentos de especificación funcional y a los “casos de uso”. Estas “historias” son escritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar. Las principales características de las historias de usuario son según (Beck, 1999): independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables.
- Plan de iteraciones: se realizan varias iteraciones sobre el sistema antes de ser entregado. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que permitan la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la iteración son: historias de usuario no abordadas, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior (Leterier, y otros, 2010).
- Plan de entregas: el plan o cronograma de entregas establece qué HU son agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma es el resultado de una reunión entre todos los actores del proyecto (cliente y desarrolladores). El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores (Joskowicz, 2008).

Diseño: la metodología XP sugiere que hay que conseguir diseños simples y sencillos, procurando hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible que a la larga costará menos tiempo y esfuerzo desarrollar. El principal artefacto generado en esta fase son las tarjetas Clase- Responsabilidad - Colaboración (CRC).

- Las tarjetas Clase-Responsabilidad-Colaboración: permiten desprenderse del método basado en procedimientos y trabajar con una metodología basada en objetos, así el programador se centra y comienza a apreciar el desarrollo orientado a objetos (Beck, 1999).

Desarrollo: consiste en la implementación o el desarrollo del código el cual se sustenta en buenas prácticas planteadas por el ciclo de vida de XP para esta etapa, entre ella se

encuentran: disponibilidad del cliente, uso de estándares, programación dirigida por las pruebas *("Test-driven programming"),* programación en pares, integraciones permanentes y ritmo sostenido. El principal artefacto que se genera en esta fase son las tareas de ingeniería o programación.

- Las tareas de programación son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, y poder ser implementadas entre uno y tres días ideales. La mayoría de las tareas de programación se derivan directamente de las historias de usuario y no tienen por qué ser comprendidas por el cliente. Cada tarea de programación será comprobada a través de los casos de prueba (Beck, 1999). Cada historia de usuario puede ser dividida en varias tareas de programación sencillas.

Prueba: XP anima a probar constantemente tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones y refactorizaciones. En XP las pruebas del sistema se dividen en dos grupos: pruebas unitarias y pruebas de aceptación (Beck, 1999).

- Pruebas unitarias: también llamadas pruebas de caja blanca, son las encargadas de verificar el código y están diseñadas por los programadores. También son llamadas pruebas modulares porque nos permiten determinar si un módulo del programa está listo y correctamente terminado (Beck, 1999).
- Pruebas de aceptación: son creadas en base a las HU, en cada ciclo de la iteración del desarrollo. El cliente debe especificar uno o diversos escenarios para comprobar que cada HU ha sido correctamente implementada. Las pruebas de aceptación son consideradas como "pruebas de caja negra". Los clientes son responsables de verificar que los resultados de estas pruebas sean correctos (Joskowicz, 2008).

1.6. Lenguaje de programación

1.6.1. JavaScript

Java Script (JS) es un lenguaje interpretado y orientado a objetos, siendo utilizado en páginas web, principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Este lenguaje hace uso de bibliotecas, una de ellas es JQuery. Esta ofrece una serie de funcionalidades basadas en JS que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes

resultados en menos tiempo y espacio. Permite simplificar la manera de interactuar con los documentos HTML, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Permite su uso en proyectos libres y privados por ser de *software* libre y código abierto. La versión a emplear es 2.1.

1.6.2. Python

Es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Es multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma por lo que se puede ejecutar sobre Windows, Linux/Unix y Mac OS, es libre de usar incluso para proyectos comerciales gracias a su licencia OSI (Iniciativa de Fuente Abierta) de código abierto. Se identifica por la identación, tipos dinámicos y gestión automática de memoria. Viene acompañado de un intérprete interactivo que permite agilizar el desarrollo de programas, pues sirve de banco de pruebas de las nuevas ideas y así conocer rápidamente el resultado. Trabaja en entornos de desarrollos con bajo consumo de recursos lo que aumenta el rendimiento de las estaciones de trabajo (Python, 2015). La versión a emplear es 2.7.

1.6.3. HTML

El lenguaje de marcas de hipertexto, HTML por sus siglas en inglés, es principalmente usado para la creación de páginas web. Se emplea para describir y traducir la estructura y la información en forma de texto, así como para ilustrar el texto con elementos como imágenes y videos. El código HTML se define mediante etiquetas (<et_inicio> </et_fin>), que permiten añadir cualquier elemento a una página web. Permite también incluir *scripts* de otros lenguajes de programación como JavaScript o *PHP*[8], para así permitir la creación de páginas dinámicas. La versión a emplear es 5.0.

1.6.4. CSS

Las Hoja de estilo en cascada, CSS por sus siglas en inglés, es un lenguaje formal que se utiliza para definir la presentación de un documento estructurado escrito en HTML o XML. El *World Wide Web Consortium* (W3C) es el organismo encargado de la especificación de CSS que sirve como estándar. La idea que se persigue es separar la estructura de un documento de su presentación. Cuando se emplea CSS, las etiquetas de los documentos HTML o *XML*[9] no proporcionan información de cómo ha de ser la

[8] PHP (acrónimo recursivo de PHP: *Hypertext Preprocessor*) es un lenguaje de código abierto especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML (Group, 2016)
[9] XML siglas en inglés de *eXtensible Markup Language* (lenguaje de marcas Extensible).

presentación, sino que únicamente marcan la estructura del documento. La correspondiente hoja de estilo se encarga de especificar cómo se ha de mostrar esa etiqueta, ejemplo de sus atributos son: color, fuente y alineación del texto (CSS, 2011). Su uso propicia el control centralizado de la presentación de la web que agiliza cualquier actuación. En cuanto a los navegadores, permite a los usuarios usar su propia hoja de estilo, aumentando la accesibilidad. La versión a emplear es 3.0.

1.7. Marcos de trabajo

1.7.1. Bootstrap

Bootstrap es un marco de trabajo o conjunto de herramientas de software libre para diseño de sitios y aplicaciones *web*. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales para adicionar. Tiene la particularidad de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño del monitor de un computador, una *Tablet* u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como *responsive design* o diseño adaptativo (ARWEB, 2014). La versión a emplear es 3.4.

1.7.2. Django

Es un marco de trabajo para el desarrollo web de código abierto escrito en Python. Se publicó bajo la licencia BSD [10]en julio del 2005 (Django, 2015).

Sus principales características son:

Mapeado objeto-relacional: las clases del modelo (base de datos) se definen en Python y se trabaja con la *API* [11]de acceso a la base de datos que provee Django. Esto permite desarrollar independientemente el motor de la base de datos y evita en cierta medida el uso de SQL.

Vistas genéricas: incorpora un sistema de vistas genéricas para hacer tareas habituales algunas de ellas son: listar registros, ver el detalle de un registro, borrar un registro.

URLs elegantes: permite crear URLs elegantes y limpias, admitiendo el uso de expresiones regulares.

Sistema de plantillas: incorpora un sistema de plantillas que permite separar el diseño gráfico y programación. Se puede editar el HTML sin tener que tocar el código Python.

[10] Es una licencia de software libre permisiva otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*).

[11] API, siglas de *Application Programming Interface*, es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas.

Cache: usa memoria cache para obtener buen rendimiento.

Aplicaciones *pluggables*[12]: las aplicaciones se pueden instalar en cualquier otro proyecto Django, es decir, son reutilizables. La versión a emplear es 1.6.

1.8. Entorno de desarrollo integrado Pycharm

Pycharm es un IDE [13]utilizado para desarrollar en el lenguaje de programación Python. Proporciona análisis de código, depuración gráfica, integración con Sistemas de Control de Versiones Distribuidas (DVCS) y soporte para el desarrollo web con Django. Es desarrollado por la empresa JetBrains y debido a la naturaleza de sus licencias tiene dos versiones, la comunidad que es gratuita, orientada a la educación y al desarrollo puro en Python, y la Professional que incluye más características como el soporte a desarrollo web (Andrearrs, 2014). La versión a emplear es 5.0.

1.9. Sistema gestor de base de datos SQLite

SQLite es un sistema de gestión de bases de datos relacional. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica, en lugar de eso, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El formato de archivo de base de datos es multiplataforma. El código es de dominio público y por lo tanto es libre para el uso para cualquier propósito, comercial o privado, siendo la base de datos de mayor despliegue en el mundo (SQlite, 2016)**.** La versión a emplear es 9.3.

1.10. Scikit-learn

Es una biblioteca libre para el lenguaje de programación Python. Cuenta con un conjunto de módulos para el aprendizaje automático y minería de datos .Entre ellos se encuentran algoritmos para:

- Clasificación: la identificación de la categoría a la que pertenece un objeto. Ejemplo Vecino más cercano.
- Regresión: la predicción de un atributo de valor continuo asociado con un objeto. Ejemplo: Cresta de regresión.

[12] Django-Pluggables proporciona un patrón de diseño para la fabricación de aplicaciones reutilizables.

[13] IDE, siglas de *Integrated Development Environment*, es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo del *software.* Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

- Agrupamiento: agrupación automática de objetos similares en conjuntos. Ejemplo: K- medias.
- Reducción de dimensionalidad: la reducción del número de variables aleatorias a tener en cuenta. Ejemplo: Factorización de la matriz no negativa.
- Selección del modelo: la comparación, evaluación y selección de parámetros y modelos. Ejemplo: Búsqueda en rejilla.
- Preprocesamiento: extracción de características y normalización. Ejemplo: Extracción de características.

Está diseñado para interoperar con las bibliotecas de Python numéricos y científicos NumPy y SciPy (developers).

NumPy: es el paquete fundamental para la computación científica con Python, que incorpora soporte para grandes conjuntos y matrices multidimensionales, junto con una gran biblioteca de alto nivel de funciones matemáticas que operan en estas matrices. Permite integrarse sin problemas y rápidamente con una amplia variedad de bases de datos. Es de código abierto y tiene muchos colaboradores. (Stefan Van Der Walt)

SciPy: es una herramienta de computación científica para Python de código abierto utilizado por los científicos, analistas e ingenieros que realizan la computación científica y computación técnica. Contiene módulos para la optimización, álgebra lineal, la integración, la interpolación, funciones especiales, procesamiento de señales e imágenes y otras tareas comunes en la ciencia y la ingeniería (SciPy.org). La versión a emplear es 1.7.

Conclusiones parciales

A partir del análisis de los referentes teóricos de los Sistemas basados en casos para apoyar el proceso de toma de decisiones, específicamente en el área de la educación superior, se arriba a las siguientes conclusiones:

- ✓ En la literatura científica consultada no se hallaron evidencias de sistemas automatizados que apoyen la toma de decisiones empleando el Razonamiento basado en casos.
- ✓ El algoritmo *Affinity Propagation* constituye una alternativa a ser empleada en el sistema basado en casos para resolver el problema de agrupamiento.
- ✓ La metodología ágil XP es adecuada para el desarrollo del sistema, ya que se adapta a las necesidades del mismo donde los requisitos son imprecisos y muy cambiantes, existiendo un alto riesgo técnico, como se evidencia en el Sistema basado en casos a desarrollar.

Capítulo II. Fases de planificación y diseño

Introducción

En el presente capítulo se precisan los elementos que conforman la propuesta de solución de la presente investigación, con el objetivo de esclarecer y comprender la misma. Se detallan los artefactos generados en la fase de planificación y diseño de la metodología de desarrollo seleccionada.

2.1. Propuesta de solución

El desarrollo de la propuesta de solución consta de 3 momentos fundamentales: modelar la Base de casos, implementar el sistema web e implementar el ciclo del RBC, los mismos se detallan a continuación.

2.1.1. Fase I: Organización de la Base de casos

En la presente investigación los **rasgos predictores** fueron seleccionados en correspondencia con las características de los estudiantes que son de interés para los profesores y están formados por el nombre, el valor y la relevancia (peso). En la siguiente tabla se detallan los rasgos, exponiendo también los valores de dominio, el tipo de las variables correspondientes y la relevancia que le es conferida.

Nombre del rasgo	**Peso**	**Valor de domino**	**Tipo**
Nota del Trabajo de Control Parcial	0.30	[1…n]	Numérico
Promedio de los resultados de las preguntas orales	0.25		
Promedio de los resultados de las preguntas escritas	0.25		
Evaluación del seminario	0.25		
Inasistencias al aula (horas/clases)	0.30		
Corte evaluativo	0.30	Bien (B) Regular (R) Mal (M)	Ordinal
Identificar conjuntos, elementos de un conjunto, el conjunto vacío, el conjunto universo, los conjuntos disjuntos, conjunto finitos e infinitos, conjuntos equivalentes.	0.1		
Representar conjuntos intencional y extensionalmente.	0.1		
Determinar la cardinalidad de un conjunto.	0.15		
Determinar relaciones de inclusión, inclusión propia e igualdad de conjuntos, a partir del concepto de conjunto y la definición de subconjunto.	0.15		

Determinar el conjunto potencia de un conjunto dado.	0.30		
Realizar cálculos de álgebra de conjunto usando las operaciones de intersección, unión, diferencia, diferencia simétrica, complemento y producto cartesiano.	0.20		
Representar gráficamente las operaciones y relaciones entre conjuntos, usando los diagramas de Venn.	0.30		
Identificar y determinar las particiones de un conjunto dado.	0.25		
Determinar las operaciones de unión, intersección y producto cartesiano de familias de conjuntos.	0.30		
Demostrar relaciones entre conjuntos mediante el empleo de las definiciones y leyes del álgebra de conjuntos, las tablas de pertenencia y/o los diagramas de Venn.	0.30		
Identificar y determinar relaciones, así como su dominio e imagen.	0.30		
Calcular la relación inversa y la composición de relaciones.	0.20		
Representar gráfica y matricialmente una relación binaria.	0.15		
Identificar las propiedades de las relaciones binarias definidas sobre un mismo conjunto y sus clasificaciones en equivalencia u orden, según dichas propiedades.	0.30		
Determinar las clases de equivalencia de una relación de equivalencia dada.	0.25		
Determinar la partición que una relación de equivalencia define para el conjunto dado (conjunto cociente).	0.20		
Obtener, dada una partición del conjunto, la relación de equivalencia correspondiente.	0.20		
Identificar si una relación de orden constituye un orden total, a partir del concepto de elementos comparables.	0.20		
Establecer un orden topológico para una relación de orden dada.	0.1		
Identificar si una relación es una función, parcial o total, y sus propiedades (inyectividad, sobreyectividad, biyectividad).	0.30		
Determinar cuándo un enunciado constituye una proposición.	0.1		
Formalizar, en el lenguaje de la lógica proposicional, enunciados del lenguaje natural.	0.3		

Expresar en el lenguaje natural fórmulas de la lógica proposicional.	0.3		
Clasificar una proposición dada en simple o compuesta.	0.1		
Determinar cuándo una expresión constituye una fórmula de la lógica proposicional a partir de las reglas de formación de fórmulas.	0.15		
Interpretar fórmulas de la lógica proposicional construyendo sus tablas de verdad.	0.20		
Clasificar una fórmula en tautología, contradicción o contingencia a partir de su tabla de verdad y/o las leyes de la lógica proposicional.	0.10		
Identificar equivalencias e implicaciones lógicas a partir del concepto de tautología.	0.20		
Demostrar equivalencias e implicaciones lógicas usando las leyes de la lógica proposicional.	0.30		
Identificar y determinar el recíproco y contra recíproco de una implicación.	0.20		
Identificar si una fórmula de la lógica proposicional está en forma normal disyuntiva (completa) o en forma normal conjuntiva (completa).	0.10		
Transformar fórmulas de la lógica proposicional a una forma normal disyuntiva (completa) o conjuntiva (completa) usando las leyes de la lógica proposicional o la tabla de verdad.	0.30		
Identificar los componentes fundamentales de los circuitos lógicos.	0.10		
Diseñar circuitos lógicos de una o más entradas con una o más salidas, dadas las proposiciones correspondientes a cada una de las salidas utilizando compuertas básicas y compuertas universales.	0.30		
Interpretar lógicamente el funcionamiento de un circuito lógico dado y extraer las proposiciones correspondientes a cada una de las salidas del mismo.	0.30		
Modelar un problema o situación y darle solución con el empleo de los circuitos lógicos.	0.30		
Utilizar las leyes de la lógica proposicional, los Mapas de Karnaugh y el Método de Quine – Mc Cluskey para simplificar circuitos lógicos.	0.30		

Formalizar razonamientos deductivos, a través de la identificación de premisas y conclusiones, usando el lenguaje de la lógica proposicional.	0.10		
Determinar la validez de un razonamiento deductivo utilizando tablas de verdad y las reglas de inferencia.	0.30		
Deducir conclusiones válidas dado un razonamiento deductivo.	0.30		
Identificar errores cometidos en una demostración dada.	0.25		
Identificar el método de demostración utilizado en la demostración de un enunciado determinado.	0.20		
Realizar demostraciones utilizando el método directo, los métodos indirectos (contra recíproco y reducción al absurdo) y el método de inducción matemática.	0.30		
Expresar enunciados del lenguaje natural usando el lenguaje de la lógica de predicados y viceversa.	0.30		
Identificar cuándo una expresión es un enunciado válido en el lenguaje de la lógica de predicados, a partir de las reglas de formación de términos o de fórmulas.	0.15		
Determinar si un enunciado es un predicado o una proposición a partir de sus definiciones.	0.20		
Interpretar fórmulas de la lógica de predicados a partir de una interpretación definida en un dominio de discurso.	0.30		
Realizar demostraciones de equivalencias lógicas a partir de las leyes de la Lógica de Predicados.	0.30		
Conocer las nociones de Teoría de la computabilidad, algoritmo, alfabeto, lenguaje y máquina de Turing.	0.20		
Diseñar una máquina de Turing para el reconocimiento de un lenguaje, representando la misma formalmente o mediante su diagrama de estados.	0.30		
Obtener un cómputo en una máquina de Turing para una cadena de entrada dada y concluir entonces si la misma es aceptada o no.	0.20		

Tabla 2 Rasgos definidos, valores de domino, tipo de variable y pesos

El peso asociado a cada rasgo fue obtenido a través del criterio de expertos, analizando para esto el nivel de importancia que cada uno le otorgaba a las características definidas. Se asignó un valor en el rango de 1 a 5, quedando finalmente la relación importancia-peso de la siguiente forma:

Importancia	**Peso(W)**
5	0.30

4	0.25
3	0.20
2	0.15
1	0.1

Tabla 3: Relación importancia-peso

El **rasgo objetivo** es un conjunto de ejercicios o una observación que desee hacerle el profesor al estudiante. Esta solución en términos de los clientes es llamado remedial.

2.1.2. Fase II: Implementación del sistema web

En esta fase se desarrollará el sistema web donde se implementarán las interfaces que facilitan la gestión de los módulos que proveerá los datos para la utilización del RBC siguiendo los siguientes pasos:

1. Definición e implementación de las clases de datos persistentes asociados a la base de datos del sistema.
2. Implementación de las clases controladoras asociadas a la base de datos del sistema y la visualización.
3. Implementación de todas las plantillas y formularios HTML.
4. Validación de la gestión de los datos asociados a los formularios.

Contenido y estructura:

Las acciones que se podrán ejecutar en común para los módulos son eliminar, inspeccionar, modificar y adicionar. El módulo Estudiante contará con la acción extra denominada “Proponer remedial” cuyo propósito es activar el ciclo del RBC para proponer los posibles remediales a aplicar.

Estudiante: módulo para gestionar los estudiantes. La información que se manipula del estudiante es: nombre, apellidos y usuario.

Registro: módulo para la gestión de las notas y desempeño de los estudiantes. La información que se manipula es: notas de los seminarios, promedio de preguntas escritas, promedio de preguntas orales, notas de los trabajos de controles parciales, cortes evaluativos e inasistencias a clases.

Habilidad: módulo para la gestión de las habilidades. La información que se manipula es la siguiente: nombre de la habilidad, descripción, peso asociado, tema y etapa a las cuales corresponde el elemento.

Tema: módulo para la gestión de los temas. La información que se manipula es: descripción del tema.

Remedial: módulo para la gestión de los remediales. La información que se manipula es: Nombre y Descripción del remedial.

Caso: módulo para la gestión de los casos. Un caso se contempla como la relación entre un estudiante con todos sus parámetros y un remedial.

2.1.3. Fase III: Ciclo de funcionamiento del RBC

En esta fase se realiza el ciclo del razonador basado en casos, la cual comprende el acceso y recuperación de los casos más semejantes y la aplicación de un procedimiento para la adaptación de las soluciones teniendo en cuenta los casos similares recuperados de la BC. Luego se cierra el ciclo del RBC cuando se incorpora el nuevo caso a la memoria de casos. Esta fase está compuesta por tres etapas, las cuales se describen a continuación.

Etapa 1: recuperación de casos semejantes

En el desarrollo de esta fase se emplearon 2 algoritmos, los cuales se detallan a continuación.

Algoritmo 1: Creación de los modelos empleando el *Affinity Propagation* [14]del Scikit-learn.

Entrada:

N x M // Matriz donde N representa la cantidad de muestras y M la cantidad de rasgos de cada muestra.

P // Nuevo caso.

Salida:

Grupo al cual pertenece el elemento P.

Pasos 1: Realizar la agrupación para determinar las etiquetas del grupo. // Se obtiene un modelo que contiene los grupos creados. Una matriz N x 1 donde se representa la etiqueta del grupo al que pertenece el elemento N.

Paso 2: Predecir el grupo más cercano al cual pertenece el nuevo elemento. // Se obtiene una matriz R x M de los elementos del grupo al cual pertenece por afinidad el elemento P.

Algoritmo 2: Recuperación de los casos más semejantes empleando función de semejanza.

Entrada: Matriz R x M, P.

Salida: Matriz R x S // Matriz de semejanza entre los elementos del grupo y el nuevo caso. S_i representa el valor de semejanza entre P y el elemento R_i.

[14] Si desea profundizar en el funcionamiento del algoritmo Affinity Propagation, consultar las bibliografías (Dueck, 2007) (descargar en www.sciencemag.orgy) y la tesis doctoral (Herranz, 2012) .

Paso 1: Calcular la función de semejanza para cada elemento del grupo con respecto al nuevo caso, para ello se emplea la ecuación:

$$\beta(P, R_i = (\sum_{i=1}^{n} W_i * \delta_i(P, R_i)) / \sum_{i=1}^{n} W_i$$

Ecuación 1: Función de semejanza entre casos

Donde P representa la información del caso nuevo y R_i representa la información del caso contenido en el grupo. La suma recorre los n rasgos que caracterizan el caso e incluye el producto del peso de cada rasgo W_i con el valor de la función de comparación por cada rasgo δ_i.

Paso 2: Calcular la similitud por rasgos.

Teniendo como base el principio de obtener la semejanza entre un estudiante existente en la base de casos y uno nuevo a analizar, los rasgos cuyo valor de dominio lo definen variables numéricas son comparados utilizando una variación de la función Manhattan ajustada. Esta permite transformar el cálculo de la distancia entre dos valores en un intervalo definido, o bien en un conjunto del cual se conozcan el mínimo y el máximo valor que puedan tomar sus elementos a un resultado que se define como semejanza. Por tanto, siguiendo la misma notación definida en la ecuación 1 la función de comparación δ i para los rasgos a emplear es:

$$\delta i(P, R_i) = 1 - \frac{| Xi(P) - Xi(R_i) |}{\max_i - \min_i}$$

Ecuación 2: Función de comparación entre rasgos

Donde $X_i(P)$ y $X_i(Ri)$ representan el valor del rasgo i de los elementos especificados, max_i y min_i representan los valores máximos y mínimos que pueden tomar los rasgos en el dominio definido respectivamente.

Los rasgos que sus valores de dominio lo definen variables de tipo ordinal discreto, expresan medidas o cualidades que no pueden ser calculadas de manera objetiva. En estos casos no son esenciales los valores en sí, sino su orden relativo. Por este motivo, son normalizados y luego comparados utilizando ecuación 2 (Cordero Morales, y otros, 2013).

Para normalizar los valores según (Xu, 2009) se prosigue de la siguiente manera:

Se define M_i como la cantidad de estados ordenados que puede tomar el rasgo X_j Se hace corresponder a cada valor del dominio, según su orden un número entero entre 1 y M_i. Para la presente investigación M_i=3; quedando los valores de los rasgos de la siguiente forma: Mal=1, Regular=2, Bien=3.

Etapa 2: adaptación y evaluación de soluciones

Estas dos fases están implícitas en una sola, para cumplir la regla del negocio de que los profesores de la asignatura tengan la posibilidad de escoger entre los cinco remediales propuestos, resultantes de la ejecución de la fase de recuperación, a su consideración el más adecuado para el nuevo problema. También se le brinda la posibilidad de crear un nuevo remedial, si considera que ninguno de los propuestos es adecuado o seleccionar uno de los existentes en el sistema.

Etapa 3: almacenamiento

En esta etapa se le delega la responsabilidad al profesor para decidir si el nuevo caso será utilizado en futuras recomendaciones y así almacenar la nueva experiencia en la memoria del sistema. Esta fase facilita la reconstrucción del conocimiento permitiendo alcanzar soluciones más precisas y cubrir un rango mayor de problemas a ser resueltos por el sistema.

2.2. Fase de planificación

2.2.1. Descripción de las Historias de Usuario

Para su confección se realizaron diferentes entrevistas a los profesores de Matemática Discreta I. Es válido destacar que si bien el cliente no fue quien escribió personalmente las HU, fue él quien diseñó su contenido, asignó las prioridades de cada una de ellas y dirigió la redacción de las mismas. El equipo de desarrollo por su parte revisó esta prioridad analizando la dependencia entre HU y asignó el costo de cada una de ellas, este se traduce en las semanas para su desarrollo. Es importante destacar, que las HU nuevas pueden describirse en cualquier momento, con esto se comprueba la flexibilidad de la metodología.

Las HU se representan mediante tablas las cuales contienen las siguientes secciones:

Código: Las siglas de HU más un número consecutivo, este permite identificar cada historia de usuario.

Nombre: Nombre que identifica la HU.

Referencia: Es el conjunto de códigos de las diferentes HU de las cuales depende actualmente la que se encuentra en desarrollo.

Prioridad: Esta característica es dada por el cliente con los valores: alta, media o baja, eligiendo aquellas historias que pueden ser utilizadas inmediatamente para entregar un apoyo útil al negocio. Se considera como altas a las HU necesarias para crear una estructura base o el esqueleto del sistema, pues de ellas se dependerá para el posterior avance del mismo. Las HU medias serán las que se centran en aportar mayor valor al negocio y las bajas las de menor contribución.

Iteración asignada: Número de la iteración en la cual se desarrollará la HU. Para el desarrollo del sistema se proponen tres iteraciones.

Puntos estimados: Tiempo estimado en semanas que se le asignará. Se tomará la unidad como una semana de trabajo.

Descripción: Breve descripción del proceso que define la HU.

Observaciones: Alguna acotación importante a señalar acerca de la HU.

Durante la fase de planificación se confeccionaron un total de 24 HU. A continuación se detalla 1 de ellas y el resto podrán ser consultadas en el anexo 1.

HISTORIA DE USUARIO	
Código: HU1	**Prioridad:** Alta
Puntos estimados:0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** _
Nombre: Adicionar usuario	
Descripción: El jefe de la asignatura escoge de la página principal la opción usuario o en el panel superior insertar usuario. Al escoger la opción nuevo usuario se le mostrará un formulario con los parámetros: usuario, contraseña, repetir contraseña. Luego de insertar correctamente estos campos, se pasa directamente a editar los datos personales: nombre, apellido, email y los permisos (activo, vista de administración, y es súper usuario) se visualiza el nuevo usuario en la lista de usuarios del sistema.	
Observaciones: El usuario y la contraseña no necesariamente tienen que ser las del dominio uci.	

Tabla 4 : HU1 Adicionar usuario

2.2.2. Requerimientos no funcionales

Los requisitos no funcionales consisten en restricciones impuestas por el entorno y tecnologías, especificaciones sobre tiempo de respuesta o volumen de información tratado por una unidad de tiempo, requisitos en cuanto a interfaces, extensibilidad y facilidad de mantenimiento (Falgueras, 2003).

A continuación se detallan los requisitos no funcionales del sistema.

Usabilidad: El sistema también podrá ser usado por aquellas personas que no poseen conocimientos avanzados en informática. Permite el acceso a cada funcionalidad de forma fácil e intuitiva. Incluye un manual de ayuda que permite establecer los conocimientos básicos para la correcta utilización del sistema.

Seguridad: Se debe garantizar que se cumplan los 3 pilares fundamentales que a continuación se detallan.

- Confidencialidad: se controla el manejo de los permisos, para que los usuarios únicamente puedan acceder a la información a la que están autorizados. Se presenta como primera acción del sistema la autenticación, cada usuario debe

ingresar su nombre de usuario único y una clave. Además cada usuario tiene un rol asignado (Administrador o Usuario normal) y cada administrador puede asignar permisos a los usuarios para que realicen determinadas acciones a las que no están autorizadas.

- Integridad: se controla el manejo de la información, para que solo aquellas personas con permiso puedan modificar la información, evitando así posibles alteraciones en los resultados de la identificación y clasificación. Solamente los usuarios con permisos de administración podrán gestionar la base de conocimientos y la información que se maneja con el sistema.
- Disponibilidad: Debe garantizarse el acceso de cada usuario del sistema las 24 horas del día.

Confiabilidad: El sistema debe guardar automáticamente la información existente en caso se algún fallo que provoque la caída del mismo.

Interfaz de usuario: Los colores a emplear serán:

- Verde para indicar acciones exitosas y en los botones Aceptar.
- Rojo para indicar acciones fallidas y en los botones Cancelar.
- Azul para indicar vínculos y *links*.
- El fondo será blanco con iconos vectorizados de color negro.

Interfaz de software:

Es necesario que el servidor este corriendo con las características siguientes:

Windows Server 2000 (o superior), Linux/Unix (cualquier distribución) o Mac OS.

Python v2.5 (o superior) y Django v1.6 (o superior).

Servidor de base de datos SQLite v9.3.1

En las computadoras clientes**:**

Un navegador web: Firefox 30.0 (o superior), Opera 8 (o superior), Chrome 6 (o superior), Internet Explorer 9 (o superior), Safari 10 (o superior).

Interfaz de *hardware*: Para un rendimiento aceptable en el uso del sistema, cada máquina cliente debe contar con:

Procesador Intel Pentium IV 2.8 GHz (o superior), o variante AMD Athlon 64x2 (o superior).

Memoria RAM de 1 GB (o superior).

2.2.3. Estimación del esfuerzo por HU

Los desarrolladores estiman cuánto esfuerzo requiere cada historia y a partir de este se define el plan de iteraciones. El cliente decide las historias que se seleccionan para cada iteración (Joskowicz, 2008).

A continuación se presenta la tabla 4 donde se resume la estimación del esfuerzo realizado por parte de las desarrolladoras y la duración total que tendrá el desarrollo del sistema propuesto. Se toma como referencia la siguiente escala:
0.1 representa un día laborable, 0.2 representa 2 días laborables, 0.3 representa 3 días laborables, 0.4 representa 4 días laborables, 0.5 representa 5 días laborables o una semana representada también por 1.0.

Historia de Usuario	**Puntos de estimación (semanas)**
UH1 Adicionar usuario	0.1
UH2 Modificar usuario	0.1
UH3 Eliminar usuario	0.1
UH4 Listar usuarios	0.1
UH5 Autenticar usuario	0.1
UH6 Modificar contraseña	0.1
UH7 Adicionar caso	0.1
UH8 Modificar caso	0.1
UH9 Eliminar caso	0.1
UH10 Listar caso	0.1
UH11 Adicionar estudiante	0.1
UH12 Modificar estudiante	0.1
UH13 Eliminar estudiante	0.1
UH14 Listar estudiante	0.1
UH15 Adicionar habilidades	0.1
UH16 Modificar habilidades	0.1
UH17 Eliminar habilidades	0.1
UH18 Listar habilidades	0.1
UH19 Adicionar remedial	0.1
UH20 Modificar remediales	0.1
UH21 Eliminar remediales	0.1
UH22 Listar remediales	0.1
UH23 Buscar estudiante	0.1
UH24 Proponer remedial	4.0
Total estimado	**10**

Tabla 5 Estimación del esfuerzo

2.2.4. Plan de iteraciones

El desarrollo del sistema fue dividido en tres iteraciones teniendo en cuenta las prioridades del cliente para maximizar el valor de negocio y la estrecha dependencia funcional entre las HU. A continuación se describen cada una de las iteraciones.

Iteración 1 e iteración 2: en estas iteraciones se persigue crear un sistema que abarca los aspectos más importantes de la arquitectura global. Esto se logrará implementando las historias de prioridad alta pues hacen referencia a la construcción de la estructura de todo el sistema. En la primera iteración se implementarán las HU comprendidas desde la 1-10 y en la segunda iteración las HU comprendidas desde la 11 hasta la 23.

Iteración 3: En esta iteración se implementará la HU de prioridad media 24 consideradas por los programadores la de mayor grado de complejidad e impacto en el negocio.

Para aproximar el tiempo de ejecución de cada iteración, se tomó como medida que cada semana consta de 5 días en los que se trabajaban 6 horas sin distracciones.

En la tabla 5 se muestra el plan de iteraciones, este incorpora el tiempo estimado para cada una de las iteraciones y las HU que se van a desarrollar.

Iteración	Historias de Usuarios	Duración total
1	Adicionar usuario	4 semanas
	Modificar usuario	
	Eliminar usuario	
	Listar usuarios	
	Autenticar usuario	
	Modificar contraseña	
	Adicionar caso	
	Modificar caso	
	Eliminar caso	
	Listar caso	
2	Adicionar estudiante	4 semanas
	Modificar estudiante	
	Eliminar estudiante	
	Listar estudiantes	
	Adicionar habilidades	
	Modificar habilidades	
	Eliminar habilidades	
	Listar habilidades	
	Adicionar remediales	
	Modificar remediales	
	Eliminar remediales	
	Listar remediales	
	Buscar estudiante	
3	Proponer remedial	4 semanas

Tabla 6: Plan de iteraciones

2.2.5. Plan de entregas

Partiendo del plan de iteraciones analizado anteriormente y en correspondencia con el mismo se realiza el plan de entregas, que se muestra en la tabla 7, en el cual se proponen 2 versiones funcionales

Iteración	Entrega	Módulo	HU que abarca	Versión
1 y 2	1: 25 abril 2016	Usuario	1-6	1.0
		Caso	7-10	
		Estudiante	11-14	
		Habilidad	15-18	
		Remedial	19-22	
		Buscador	23	
3	2: 15 junio 2016	Razonador	24	2.0

Tabla 7: Plan de entrega.

2.3. Fase de diseño

2.3.1. Patrón de arquitectura del sistema

Django se basa en el Modelo Vista Controlador (MVC), aunque sus desarrolladores prefieren llamarlo Modelo Plantilla Vista (MPV). El controlador pasa a ser la vista y la vista pasa a denominarse plantilla. En Django, una vista describe los datos que se ofrecen al usuario pero no necesariamente su aspecto. Una vista habitualmente delega los datos a una plantilla que describe la forma de presentarlos. Se dice que el controlador de un patrón MVC clásico estaría representado por el propio *framework*.

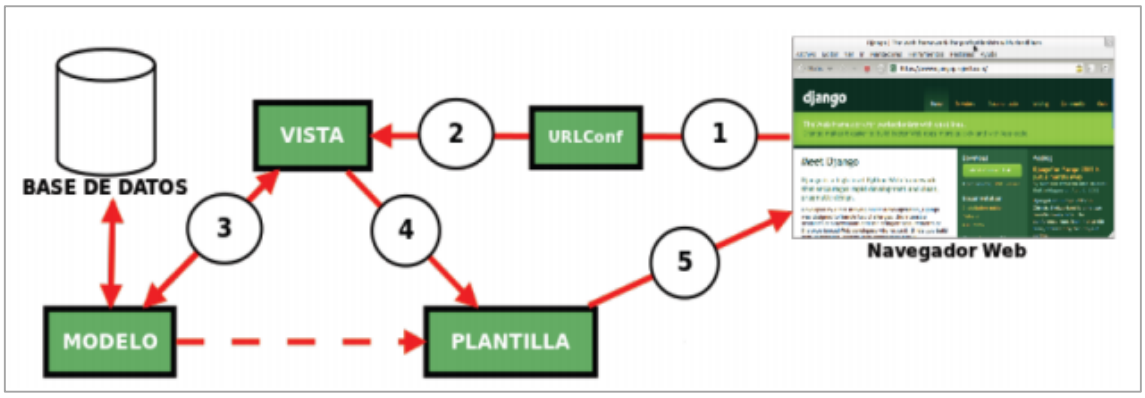

Figura 1: Funcionamiento del MPV de Django. Fuente: Sergio Infante Montero

1. El navegador manda una solicitud.
2. El URLConf interpreta la solicitud y ubica la vista apropiada.
3. La vista interactúa con el modelo para obtener los datos.
4. La vista selecciona la plantilla apropiada.
5. La plantilla renderiza la respuesta a la solicitud del navegador.

2.3.2. Patrones de diseño:

“Un patrón es una descripción de un problema y la solución a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias” (Larman, 1999).Resulta una buena práctica antes de elaborar los diagramas de interacción asignar correctamente las responsabilidades. Los patrones de diseño constituyen entonces, el auxiliar principal para esta tarea.

Patrones GRASP

Los Patrones de *Software* para la Asignación General de Responsabilidad (GRASP), describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones (Larman, 1999).En la implementación de la aplicación se utilizan los siguientes:

- **Patrón Creador:** encargado de la creación de instancias. Es uno de los patrones más comunes en un sistema orientado a objetos. Se pone de manifiesto al utilizar las class-based-views [15] de Django.
- **Patrón Experto:** asigna la responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Esto se evidencia en las clases que modelan los datos persistentes del sistema. Ejemplo: la clase Estudiante tiene la responsabilidad de crear las instancias de estudiantes pues contiene la información necesaria para ello.
- **Patrón Bajo Acoplamiento**: asigna responsabilidades a las clases para mantener bajo acoplamiento. Explica cómo dar soporte a una dependencia escasa y a un aumento de la reutilización. Se evidencia al utilizar los modelos de datos persistentes en la capa de presentación de Django.
- **Patrón Alta Cohesión:** asigna responsabilidades de forma que la cohesión entre las clases siga siendo alta. Explica cómo mantener la complejidad dentro de los límites manejables.
- **Patrón Controlador:** asigna la responsabilidad del manejo de los eventos del sistema a una clase que represente un sistema global. Es apreciable en las vistas, las cuales controlan la interacción visual de la información.

Patrones GoF

Los patrones Banda de Cuatro (GoF), complementan a los patrones GRASP y en ocasiones se puede encontrar una contraposición entre este tipo de patrones, e incluso,

[15] Las class-based-views son vistas que permiten realizar de forma sencilla, las funcionalidades básicas en casi todos los sistemas (Crear, Actualizar, Eliminar, Listar).

podría inferirse que muchos de los patrones GoF son variantes de los patrones GRASP (Pressman, 2002). En la implementación de la aplicación se utiliza el siguiente:

Patrón Mediator: encargado de manejar la interacción entre los diferentes subsistemas. Se evidencia en el mapeo objeto relacional a cargo del motor de Django.

2.3.3. Tarjetas clase-responsabilidad-colaboración (CRC)

A continuación se presenta la tarjeta CRC Remedial, el resto pueden ser consultadas en el anexo 2.

TARJETA CRC	
Clase: Remedial	
Descripción: Gestiona la información de los remediales del sistema	
Atributos:	
Nombre	**Descripción**
ID	Identificador único del remedial
Descripción	Contenido del remedial
Tema	Tema al que pertenece
Cantidad	Total de remediales existentes
Responsabilidad: • Crear remedial • Eliminar remedial • Modificar remedial • Listar remedial • Determinar cantidad	**Colaboración:** • Caso • Tema • Estudiante

Tabla 8: CRC-Remedial

Conclusiones parciales

La ejecución de las fases de planificación y diseño permitieron:

- ✓ Determinar las características funcionales del sistema representadas en historias de usuario para su comprensión y desarrollo.
- ✓ Conformar el plan de iteraciones para establecer en correlación con el cliente, las historias de usuario que serán implementadas en cada iteración y estimar la duración del sistema a desarrollar, favoreciendo la correcta organización del mismo.
- ✓ Confeccionar las tarjetas CRC para determinar la relación funcional entre las clases del sistema y las funcionalidades que serán implementadas por cada clase.

Capítulo III. Fase de desarrollo y pruebas

Introducción

En este capítulo se describen los artefactos generados en la implementación del sistema y las pruebas hechas al mismo. Se detallan las tareas de implementación en las que se fragmenta cada HU que se implementará. Posteriormente se elaboran los casos de prueba.

3.1. Fase de Desarrollo

3.1.1. Tareas de ingeniería de desarrollo

Para determinar las tareas que componen a una HU se realiza una reunión con todos los miembros del equipo de desarrollo y se obtiene el listado de las mismas, como propusiera Kent Beck y Martin Fowler en el libro Planning Extreme Programming en su edición del 2000.

A continuación se relacionan las tablas correspondientes a las tareas de programación realizadas, cuyos campos responden a las siguientes descripciones:

Número historia de usuario: Número de la HU a la que corresponde. Índice de la HU a la que se corresponde esta tarea de programación.

Número tarea: Índice de la tarea de programación. Es un número único que se le asigna a cada tarea de programación que pertenece a una HU determinada con el fin de lograr una mejor organización de éstas.

Nombre de tarea: Nombre de la tarea de programación. Debe ser descriptivo, en la medida de las posibilidades, de lo que se realizará y no muy extenso.

Tipo de tarea: Informa el tipo de tarea a realizar. Pueden ser:

- Desarrollo: tarea que se realizará por primera vez.
- Corrección: tarea que se haya realizado a partir de una anterior incorrectamente, es decir no supero todos los casos de prueba que le corresponden.
- Mejora: tarea que se realiza a partir de una anterior que se realice correctamente pero se incorporan nuevos requerimientos para la misma, ahora tendrá que ser modificada para pasar satisfactoriamente los nuevos casos de prueba adicionados.

Puntos estimados: representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizara para su realización.

Fecha inicio: fecha estimada de inicio de realización.

Fecha fin: fecha estimada de fin de realización.

Descripción: se describe en qué consiste la tarea y qué elementos deben cumplirse para declarar la tarea terminada.

A continuación se presenta la tarea de programación “Creación del modelo usuario”, el resto podrán ser consultadas en el anexo 3.

TAREA DE PROGRAMACIÓN

No. de tarea: 1	**No.de HU: 1**
Nombre de la tarea: Creación del modelo Usuario	
Tipo de tarea: Desarrollo	**Responsable:**
Fecha inicio: 16-2-2016	**Fecha fin:**18-2-2016
Descripción: Se crea el modelo encargado de persistir la información referente al usuario en la base de datos.	

Tabla 9: TP1 Creación del modelo Usuario

3.2. Fase de Prueba

3.2.1. Pruebas de aceptación

Como criterio de aprobación de cada iteración se tomó que el 100% de los casos de prueba sean exitosos para pasar de iteración. El objetivo de estas pruebas no es tener un conjunto de casos escritos que cubran el 100% del código, sino poder realizarle pruebas al sistema desde el punto de vista del usuario. Para la realización de cada una de las pruebas de aceptación se siguieron una serie de pasos que se muestran a continuación:

1. Identificar todas las acciones en la HU.
2. Para cada acción escribir al menos una prueba.
3. Para algunos datos, reemplazar las entradas que hacen que la acción ocurra y llenar en la casilla resultados esperados los resultados obtenidos.
4. Para otros datos, reemplazar las entradas que hacen que la acción falle, y registrar los resultados.

Para representar las pruebas de aceptación se definieron los siguientes elementos:
Código: representa al caso de prueba, incluye el número de HU, de la prueba y si posee diferentes escenarios.
HU: número de la HU a la cual pertenece.
Nombre: junto al código conforma el identificador del caso de prueba.
Descripción: acción que debe realizar el sistema.
Condiciones de ejecución: describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
Pasos de ejecución: pasos para realizar el caso de prueba.
Resultados Esperados: descripción de la respuesta del sistema ante el caso de prueba.
Resultado Obtenido: respuesta visual del sistema después de realizar el caso de prueba.
Evaluación de la prueba: clasificación de la prueba en satisfactoria o insatisfactoria.

3.2.2. Pruebas de aceptación en la primera iteración

PRUEBA DE ACEPTACIÓN	
Código: UH11_P2	**HU:** 11

Nombre: Adicionar estudiante
Descripción: El administrador del sistema una vez accedido al mismo luego de ser autenticado, seleccionará del menú la opción “Insertar estudiante”. Se le mostrará una ventana con los datos requeridos para realizar la operación, luego de ser insertados los mismos, el sistema realizará su validación. De no ser correctos el sistema debe mostrar un mensaje de error y no realizar la operación de inserción.
Condiciones de ejecución: Usuario autenticado con permisos de administración
Pasos de ejecución: 1. El administrador se autentica en el sistema. 2. Del menú principal seleccionará “Estudiante”- “Insertar estudiante”. 3. Se le mostrará una pantalla solicitando los parámetros nombre, apellidos, usuario promedio de las notas alcanzadas en las preguntas orales, en las escritas y la nota alcanzada en la habilidades evaluadas hasta ese momentos. 4. El administrador introduce parámetros incorrectos y presiona el botón “Aceptar”. 5. El sistema verifica los parámetros. 6. El sistema muestra un mensaje de error.
Resultados esperados: El sistema mostrará un mensaje de error y no realizará la inserción del remedial en la base de datos.
Resultado obtenido: El sistema mostró un mensaje de error pero realizó la inserción del estudiante en la base de datos.
Evaluación de la prueba: Insatisfactoria

Tabla 10: Pruebas de aceptación Adicionar estudiante

3.2.3. Pruebas de aceptación en la segunda iteración

PRUEBA DE ACEPTACIÓN	
Código: UH23_P2	**HU: 23**
Nombre: Adicionar remedial	
Descripción: El administrador del sistema una vez accedido al mismo luego de ser autenticado, seleccionará del menú la opción “Remedial” -“Crear remedial”. Se le mostrará una ventana con los datos requeridos para realizar la operación, luego de ser insertados los mismos, el sistema realizará su validación. De ser correctos se inserta en el sistema.	
Condiciones de ejecución: Usuario autenticado con permisos de administración	
Pasos de ejecución: 1. El administrador se autentica en el sistema. 2. Del menú principal seleccionará “Remedial”- “Insertar remedial”. 3. Se le mostrará una pantalla solicitando nombre, y descripción.	

4. El administrador introduce parámetros incorrectos y presiona el botón “Aceptar”. 5. El sistema verifica los parámetros y señala los campos incorrectos en rojo y un mensaje de error.
Resultados esperados: El sistema señala los parámetros incorrectos, muestra un mensaje de error y no realizará la inserción del remedial en la base de datos.
Resultado obtenido: El sistema mostró un mensaje de error y no realizó la inserción del remedial en la base de datos.
Evaluación de la prueba: Satisfactoria

Tabla 11: Prueba de aceptación Adicionar remedial.

3.2.4. Pruebas de aceptación en la tercera iteración

PRUEBA DE ACEPTACIÓN	
Código: UH23-P1	**HU:** 23
Nombre: Proponer remedial	
Descripción El profesor luego de ser autenticado, seleccionará del menú la opción “Proponer remedial”. Se le mostrará una ventana con los datos requeridos para realizar la operación, luego de ser insertados los mismos, el sistema realizará la correspondiente validación. De ser correctos el sistema le dará como respuesta una lista de remediales.	
Condiciones de ejecución : Profesor autenticado	
Pasos de ejecución: 1. El profesor se autentica en el sistema. 2. Del menú principal seleccionará “Estudiante”- del estudiante especifica etapa del curso en la que se encuentra (1,2 ó 3). 3. Se le mostrará una pantalla solicitando las evaluaciones para las habilidades que componen dicha etapa y las notas del TCP, promedio de preguntas escritas realizadas, de las orales, el seminario e inasistencias a clases. 4. El administrador introduce parámetros correctos y presiona el botón “Remedial”. 5. El sistema verifica los parámetros. 6. El sistema devuelve un listado de remediales que corresponden a estudiantes de cursos precedentes que presentan características semejantes a la descripción insertada. 7. El profesor elige un remedial de la lista propuesta o puede insertar uno nuevo. 8. Luego de seleccionado el remedial presiona el botón aceptar. 9. El sistema indica el éxito de la operación mediante un mensaje.	
Resultados esperados:	

El sistema devolverá un listado de remediales para seleccionar y la opción de crear un nuevo remedial. El profesor puede elegir entre las dos opciones.
Resultado obtenido: El sistema devuelve un listado de remediales pero no permite al profesor crear un nuevo remedial
Evaluación de la prueba: Insatisfactoria

Tabla 12: Pruebas de aceptación Proponer remedial.

3.2.5. Resumen de los resultados obtenidos

Se realizaron tres iteraciones de pruebas, detectándose de forma general 19 no conformidades, el siguiente gráfico muestra el resultado obtenido:

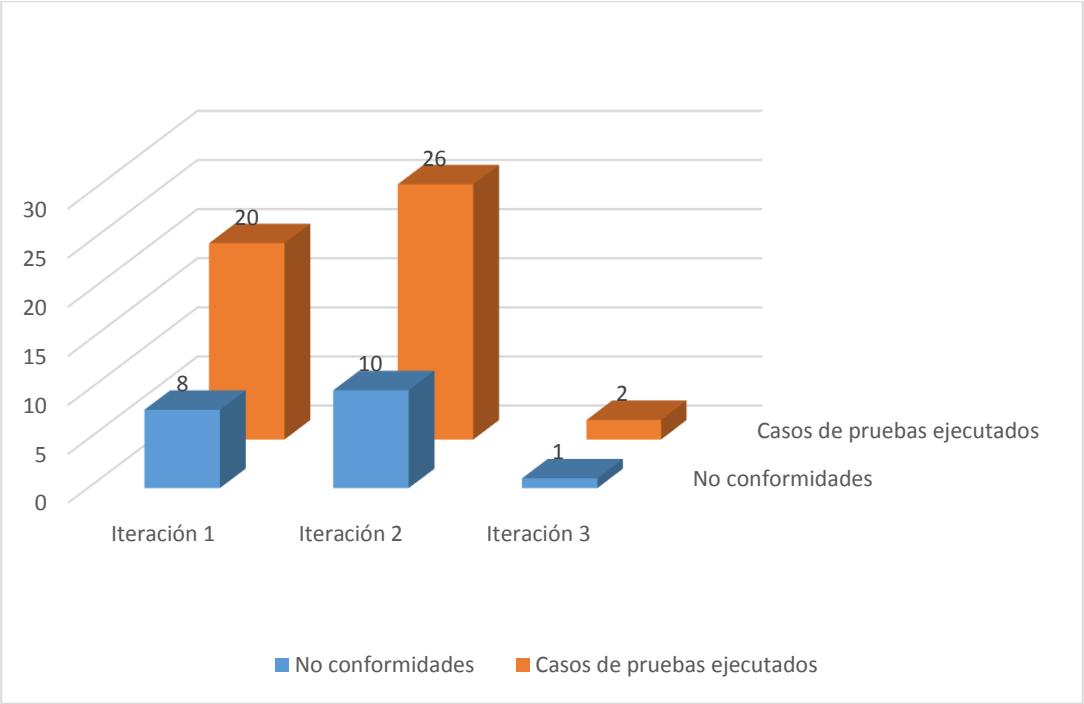

Ilustración 2: Resumen de no conformidades detectadas en cada iteración

Iteración 1: Se ejecutaron 20 casos de pruebas y se detectaron 8 no conformidades.

Iteración 1: Se ejecutaron 26 casos de pruebas y se detectaron 10 no conformidades.

Iteración 1: Se ejecutaron 2 casos de pruebas y se detectó 1 no conformidad.

3.2.6. Pruebas unitarias

Para automatizar las pruebas unitarias se empleó framework de pruebas automatizadas (PyUnit) que Django incorpora. Las mismas se escriben y ejecutan desde un archivo que se crea desde el inicio del proyecto llamado “tests.py”. Los test se crean a medida que se hacen cambios en la aplicación, se puede verificar que el código todavía funciona como estaba originalmente pensado, sin tener que usar tiempo para hacer pruebas de forma manual.

Se realizaron 15 pruebas a las funcionalidades, enfocadas en probar las *urls* y las acciones: crear, eliminar, modificar, listar e inspeccionar de los módulos, previendo que no se hicieran inserciones

dobles y garantizar que las vistas utilizan correctamente los *templates*. A continuación se muestran evidencias de las pruebas realizadas a las acciones registro, el resto puede consultarse en el anexo 5.

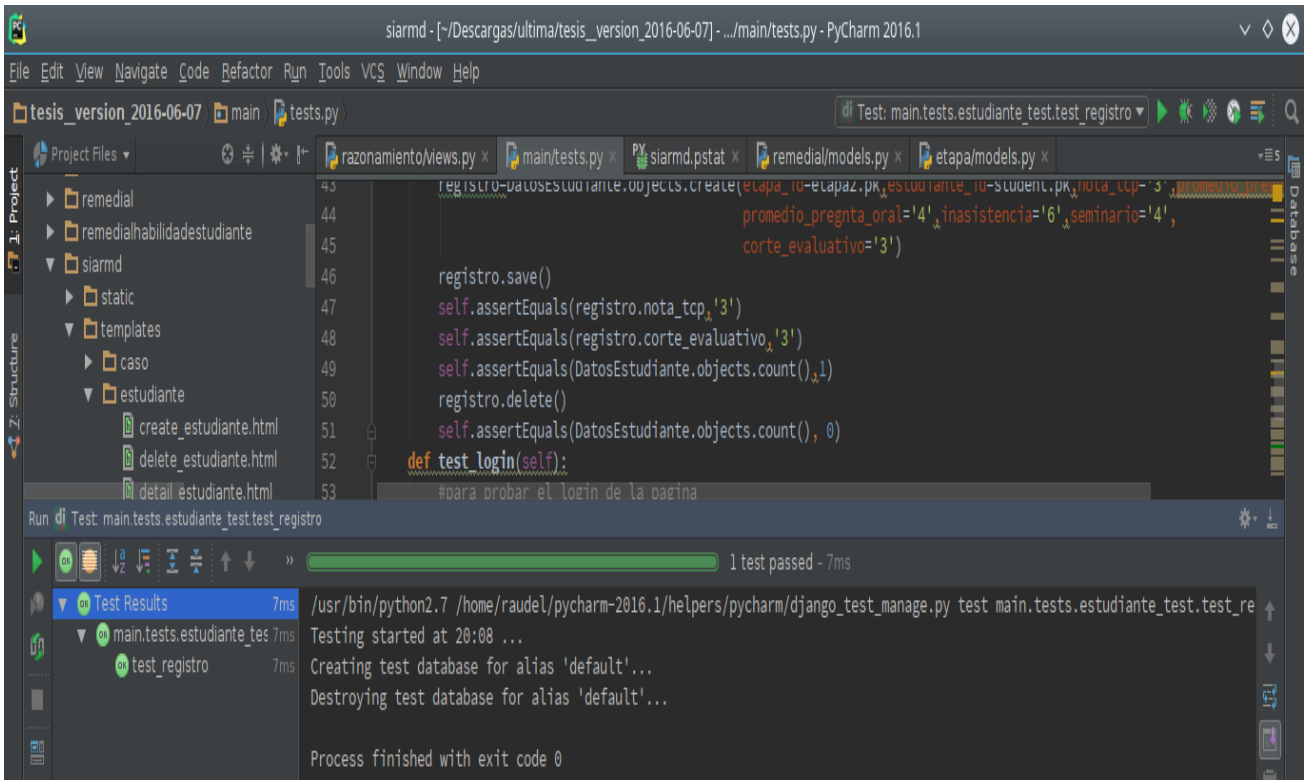

Figura 4: Prueba para acciones con registros.

Conclusiones parciales

La ejecución de las fases de desarrollo y prueba permitieron:

- ✓ Implementar las tareas de ingeniería en correspondencia con los requerimientos funcionales descritos, para obtener las correspondientes versiones funcionales del producto.
- ✓ Realizar las pruebas de aceptación para asegurar que las funcionalidades del sistema cumplieran con las especificaciones descritas en las HU. Las mismas le permitieron al cliente determinar cuan funcional estaba el sistema y al equipo de desarrollo evaluar el nivel de éxito en la implementación durante una iteración.
- ✓ Las pruebas unitarias permitieron verificar que los módulos (unidad de código) estuviesen funcionales, propiciando llegar a la fase de integración con la seguridad de que el código estaba funcionando correctamente. Además facilitó al equipo de desarrollo asegurarse que los cambios en el código no hayan introducido errores.

Conclusiones generales

La investigación realizada se logra como producto final un Sistema basado en casos que apoya la toma de decisiones en la asignación de remediales, dando cumplimiento con el objetivo general trazado, arribándose a las siguientes conclusiones:

- ✓ En la etapa de recuperación de los casos es viable la utilización del algoritmo de agrupamiento *Affinity Propagation* para generar un modelo de similitud entre elementos y reducir el espacio de búsqueda.
- ✓ El conocimiento de los expertos en el área de la asignatura Matemática Discreta I correspondientes al trabajo didáctico con los estudiantes de bajo aprovechamiento académico, es modelado en una base de casos que otorga disponibilidad a las experiencias para el posterior empleo en sistemas de Razonamiento basado en casos.
- ✓ Las pruebas aplicadas a la propuesta de solución propiciaron una correcta validación de las funcionalidades y el diseño del sistema de Razonamiento basado en casos para la propuesta de remediales.

Recomendaciones

Se proponen como futuros trabajos en esta área de investigación, incorporar a la herramienta las siguientes funcionalidades:

- ✓ Crear una interfaz de usuario para cargar masivamente la base de casos a través de un archivo con extensión CSV.
- ✓ Integrar la aplicación con el sistema de gestión universitaria de la UCI para que se pueda extender el uso del sistema a otras asignaturas.

Bibliografía

Ferriol Ortiz, Acralus y Azahares Reyes, Enmanuel . 2009. *Análisis y diseño del Módulo Registro y Control Docente para Akademos v2.0.* 2009.

Kolodner, Janet y Kaufmann, Morgan . *Case–Based Reasoning.*

Pulla Quezada, Oswaldo y Vergara Argudo, Ricardo . 2011. *Implementación de un Sistemas de Gestión Académica como soporte a la Toma de Decisiones, para el análisis, evaluación, seguimiento y control actividades docentes.* 2011.

Andrearrs. 2014. PyCharm el IDE de Python. [En línea] 10 de junio de 2014. http://hipertextual.com/archivo/2014/06/pycharm-ide-python/.

Araujo, Basilio Sierra. 2006. *Aprendizaje Automático: conceptos básicos y avanzados .* 2006.

Arjona, Manuel Miguel Giménez. 2006. *Estudio para la implementación de un sistema de razonamiento basado en casos.* 2006.

ARWEB. 2014. ¿Qué es Bootstrap y cómo funciona en el diseño web? [En línea] 26 de septiembre de 2014. http://www.arweb.com/chucherias/editorial/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web.htm.

Ayala, Alejandro Peña. 2006. *Sistemas basados en conocimiento: Una base para su concepción y desarrollo.* 2006.

Beck, Kent. 1999. *Extreme Programming Explained.* 1999.

Collera, Alfonso. 2005. *Sistema de Gestión Académica:Módulo de Control Docente.* 2005.

Cordero Morales, Dasiel, Ruiz Constanten, Yadira y Torres Rubio, Yoanny. 2013. *Revista Cubana de Ciencias Informáticas:Sistema de Razonamiento Basado en Casos para la identificación de riesgos de software.* [Revista vol.7 no.2] La Habana : s.n., 2013. ISSN 2227-1899.

CSS. 2011. Cascading Style Sheets (CSS) Snapshot 2010. [En línea] 12 de mayo de 2011. http://www.w3.org/TR/css-2010/#css.

developers, scikit-learn. scikit-learn. *scikit-learn.* [En línea] http://scikit-learn.org/stable/about.html.

Django. 2015. Django Project. [En línea] 2015. www.djangoproject.com.

Dueck, Delbert. 2007. *Clustering by Passing Messages Between Data Points.* 2007.

Eguiluz, Javier. 2006-2016. LibrosWeb. *Introducción a JavaScript.* [En línea] 2006-2016. http://librosweb.es/libro/javascript/capitulo_1.html.

Gálvez Lio, Daniel. 1998. *Curso de sistemas basados en el conocimiento.* Las Villas : s.n., 1998.

Giménez Arjona, Manuel. 2006. *Estudio para la implementación de un sistema de razonamiento basado en casos.* 2006.

Gross, Thomas R. 2000. *Framework Design: A Role Modeling Approach.* 2000.

Group, The PHP. 2016. PHP. *PHP.* [En línea] 28 de mayo de 2016. http://php.net/manual/es/intro-whatis.php.

Gutierrez Rivero, Yandry. 2015. *GECMA.* La Habana, Cuba : s.n., 2015.

Hanney, Kathleen y Keane, Mark . 1995. *Systems, tasks,and adaptation knowledge: Revealing some dependencies.* 1995.

Hellriegel y Slocum. 2004. 2004.

Herranz, María del Soto Montalvo. 2012. *Estudio y nuevas estrategias en el uso de las Entidades Nombradas en el Clustering Bilingüe de noticias.* 2012.

Joskowicz, Jose. 2008. *Reglas y prácticas en Extreme Programing.* Universidad de Vigo.España : s.n., 2008.

Kolodner. 1992. *An Introduction to Case-Based Reasoning.* 1992.

Larman, Craig. 1999. *UML y Patrones Introducción al análisis y diseño orientado a objetos y al proceso.* 1999.

Leterier, Patricio y Penádes, María Carmen. 2010. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Camino de Vera, Valencia : Universidad Politécnica de Valencia, 2010.

Montes, Miguel Cárdenas. 2013. *Gráficas estadística y minería de datos con python.* 2013.

Oralla, José Hernández. 2004. *Introducción a la Minería de Datos.* 2004.

Overview, Product. 20013. *Database Management System.* 20013.

Patón, Dr. Eduardo Fernández Medina. 2006-2007. INGENIERÍA DEL SOFTWARE /Cuarto Curso. [En línea] 2006-2007. http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf..

Paul, Christopher. 2002. *The Linux Development Platform: Configuring, Using and Maintaining a Complete Programming Environment.* 2002.

Per Tutatis. 2016. Per Tutatis! [En línea] 2016. [Citado el: 16 de febrero de 2016.] http://pertutatis.cat/que-son-los-sistemas-de-apoyo-a-la-toma-de-decisiones-dds/.

Pintos, Salvador y Queipo. 2001. *Fundamentos de Data Mining y sus Aplicaciones en la Gerencia Integrada de Yacimientos.* 2001.

Pressman, Roger. 2002. *Ingeniería de Software. Un enfoque práctico.* 2002.

programación, Lenguajes de. Lenguajes de programación. [En línea] http://www.lenguajes-de-programacion.com/.

Python. 2015. [En línea] 2015. www.python.org.

Ramírez Blanco, Keilyn y Palacios Palacios, Ana Gabriela . 2013. *Gestión del proceso de toma de decisiones en el Colegio Técnico Profesional de General Viejo.* 2013.

Russell, Stuart y Norving, Peter. *Inteligencia artificial: Un enfoque moderno.*

SciPy.org. SciPy.org. [En línea]

Serie Científica de la Universidad de las Ciencias Informáticas:Aplicando el método de Boehm y Turner. **Velázquez, Mairelys Boeras, Barroso, Laritza Cabrera y Castro, Eileén Llano.** s.l. : Ediciones Futuro.

Solís, Manuel Calero. 2003. *Una explicación de la programación extrema (XP).* 2003.

SQlite, Comunidad. 2016. SQlite. [En línea] 2016. https://www.sqlite.org/docs.html.

Stefan Van Der Walt, S. Chris Colbert, Gaël Varoquaux. *The NumPy array: a structure for efficient numerical computation.*

Stoner. 2003. 2003.

Stuart , Russell y Norvig, Peter . 2004. *Inteligencia Artificial : Un enfoque moderno.* Madrid : Pearson Educación, 2004.

Tsatsoulis, Costas. 1989. *Case-Based Design And Learning In Telecommunications, Department of Electrical and Computer Engineering.* 1989.

UCI. 2002. Modelo del profesional de la UCI. *Intranet.* [En línea] 2002. http://intranet2.uci.cu/sites/default/files/pdf_formacion/Modelo%20del%20Profesional%20y%20Objetivos%20Generales.pdf.

WANG, JUNYING,XINNA. 2007. *Adaptive Affinity Propagation Clustering.* 2007.

Xu, Rui. 2009. *Clustering.* 2009.

Anexos

Anexo 1 Historias de usuarios

HISTORIA DE USUARIO	
Código: HU2	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** HU1,HU4
Nombre: Modificar usuario	
Descripción: El jefe de la asignatura escoge de la página principal la opción usuario o en el panel superior el listar usuarios. Se visualiza el listado de los usuarios que existen en el sistema. Se selecciona el usuario que desea reformar y se escoge la acción actualizar datos de perfil. Se le muestra un formulario con los parámetros: nombre, apellidos, email, y los permisos (activo, vista de administración, y es súper usuario). Luego de guardar esos datos, se regresa a la lista de usuarios del sistema.	
Observaciones: Para poder modificar un usuario debe estar previamente insertado en el sistema.	

Tabla 13: HU2 Modificar usuario

HISTORIA DE USUARIO	
Código: HU3	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** HU1,HU4
Nombre: Eliminar usuario	
Descripción: El jefe de la asignatura escoge de la página principal la opción usuario o en el panel superior el listar usuarios. Se visualiza el listado de los usuarios que existen en el sistema. Se selecciona el usuario que desea eliminar y se escoge la acción eliminar datos de perfil. Se debe verificar que verdaderamente desea realizar la operación a través de una interfaz de confirmación. Luego realizada la eliminación se visualizarán los cambios en la lista de usuarios del sistema.	
Observaciones: Para poder eliminar un usuario debe estar previamente insertado en el sistema.	

Tabla 14: HU3 Eliminar usuario

HISTORIA DE USUARIO	
Código: HU4	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:**
Nombre: Listar usuarios	
Descripción: El jefe de la asignatura escoge de la página principal la opción usuario o en el panel superior el listar usuarios. Se visualiza el listado de los usuarios que existen en el sistema, además de brindarle	

la opción crear usuario y las acciones: ver datos de perfil, actualizar datos de perfil, y eliminar datos de perfil.
Observaciones: Para poder listar usuarios deben estar previamente insertados en el sistema.

Tabla 15: HU4 Listar usuarios

HISTORIA DE USUARIO	
Código: HU5	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Ofelia Peláez Soto	**Referencias:**
Nombre: Autenticar usuario	
Descripción: El profesor para ingresar en el sistema debe autenticarse. La página de autenticación le mostrará un formulario con los campos usuario y contraseña. Luego de ser completados y verificados los parámetros requeridos accede al sistema teniéndose en cuenta el rol y los permisos otorgados.	
Observaciones:	

Tabla 16: HU5 Autenticar usuario

HISTORIA DE USUARIO	
Código: HU6	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Ofelia Peláez Soto	**Referencias:**
Nombre: Modificar contraseña	
Descripción: El profesor puede modificar su contraseña en cualquier momento solo necesita escoger la opción cambiar contraseña en el panel superior de la página principal. Se muestra un formulario con los campos: contraseña actual, nueva contraseña, repetir la nueva contraseña. Luego de validar estos datos, se guardan y se retorna a la página principal del sistema.	
Observaciones: Para poder cambiar la contraseña el profesor debe estar autenticado en el sistema.	

Tabla 17: HU6 Modificar contraseña

HISTORIA DE USUARIO	
Código: HU19	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:**
Nombre: Listar habilidades	
Descripción: El profesor escoge de la página principal la opción habilidad o en el panel superior en el desplegable de habilidades y se escoge la opción listar habilidades. Se visualiza el listado de los habilidades registrados en el sistema. Se selecciona el que se desea eliminar y se escoge la acción eliminar datos estudiantes. Se debe verificar que verdaderamente desea realizar la operación a través de	

una interfaz de confirmación. Luego realizada la eliminación se visualizarán los cambios en la lista de habilidades del sistema.
Observaciones: Para listar las habilidades debe haber sido insertadas previamente en el sistema.

Tabla 18: HU19 Listar habilidades

: HHISTORIA DE USUARIO	
Código: HU20	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:**
Nombre: Adicionar remedial	
Descripción: El profesor de la asignatura escoge la opción remedial en la página principal o en el panel superior en la lista desplegable de remediales se selecciona la opción crear nuevo remedial. Se visualizará un formulario para inserta los parámetros: nombre y descripción. Luego de insertar correctamente estos campos, se visualiza el nuevo remedial en la lista de remediales del sistema.	
Observaciones:	

Tabla 19: HU20 Adicionar remedial

HISTORIA DE USUARIO	
Código: HU21	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** HU20
Nombre: Modificar remedial	
Descripción: El profesor escoge de la página principal la opción remedial o en el panel superior en el desplegable de remediales y se escoge la opción listar remediales. Se visualiza el listado de las remediales que existen en el sistema. Se selecciona los remediales que desea reformar y se escoge la acción actualizar datos de los remediales. Se le muestra un formulario con los parámetros: nombre y descripción. Luego de guardar esos datos, se regresa a la lista de remediales del sistema.	
Observaciones: Para modificar un remedial debe estar insertado previamente en el sistema	

Tabla 20: HU21 Modificar remedial

HISTORIA DE USUARIO	
Código: HU22	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** HU20
Nombre: Eliminar remedial	
Descripción: El profesor escoge de la página principal la opción remediales o en el panel superior en el desplegable de remediales y se escoge la opción listar remediales. Se visualiza el listado de los remediales registrados en el sistema. Se selecciona el que se desea eliminar y se escoge la acción	

eliminar datos del remedial. Se debe confirmar que verdaderamente se desea realizar la operación a través de una interfaz de confirmación. Luego realizada la eliminación se visualizarán nuevamente la lista de remediales.
Observaciones: Para eliminar un remedial debe estar insertado previamente en el sistema

Tabla 21: HU22**:** Eliminar remedial

HISTORIA DE USUARIO	
Código: HU23	**Prioridad:** Alta
Puntos estimados: 0.1	**Iteración asignada:** 1
Responsable: Yeinelis Hierrezuelo Ramírez	**Referencias:** HU20
Nombre: Listar remediales	
Descripción: El profesor escoge de la página principal la opción remediales o en el panel superior en el desplegable de remediales y se escoge la opción listar remediales. Se visualiza el listado de los remediales registrados en el sistema. Se selecciona el que se desea eliminar y se escoge la acción eliminar datos remediales. Se debe verificar que verdaderamente desea realizar la operación a través de una interfaz de confirmación. Luego realizada la eliminación se visualizarán los cambios en la lista de remediales del sistema.	
Observaciones: Para listar los remediales deben estar previamente insertados en el sistema.	

Tabla 22: HU23 Listar remediales

Anexo 2 Tarjetas CRC

TARJETA CRC	
Clase: Estudiante	
Descripción: Gestiona la información de los estudiantes del sistema	
Atributos:	
Nombre	**Descripción**
Nombre y apellidos	Nombre completo del estudiante
Usuario	Usuario del estudiante que es el identificador
Responsabilidad: • Crear estudiante • Eliminar estudiante • Modificar estudiante • Listar estudiante • Determinar cantidad de estudiantes	**Colaboración:** • Estudiante-datos

Tabla 23: CRC Estudiante

TARJETA CRC
Clase: Razonador
Descripción: Realiza el razonamiento basado en casos del sistema

Atributos:	
Nombre	**Descripción**
Responsabilidad:	**Colaboración:**
Clasificar estudiante	Caso, Estudiante
Proponer remedial	
Incorporar remedial al sistema	

Tabla 24: CRC Razonado

TARJETA CRC	
Clase: Caso	
Descripción: Gestiona la información de los casos del sistema	
Atributos:	
Nombre	**Descripción**
Remedial-id	Identificador único del caso
Estudiante-datos-id	Identificador único de estudiante-datos
Responsabilidad: • Crear caso • Eliminar caso • Modificar caso • Listar casos	**Colaboración:** • Remedial • Estudiante-Datos

Tabla 25: CRC Caso

TARJETA CRC	
Clase: Habilidad	
Descripción: Gestiona la información de las habilidades del sistema	
Atributos:	
Nombre	**Descripción**
ID	Identificador único de la habilidad
Tema-id	Tema al que está asociada a la habilidad
Descripción	Descripción de la habilidad
Etapa-id	Etapa asociada a la habilidad
Peso	Peso asociada a la habilidad
Responsabilidad:	**Colaboración:**
Crear habilidad	Tema, Estudiante-Datos, Etapa
Eliminar habilidad	
Modificar habilidad	
Listar habilidades	

Tabla 26: CRC Habilidad

TARJETA CRC	
Clase: Etapa	
Descripción: Gestiona la información de las etapas del sistema	
Atributos:	
Nombre	**Descripción**
ID	Identificador único de la etapa
Descripción	Descripción de la etapa
Responsabilidad:	**Colaboración:**
Crear etapa	Habilidad
Eliminar etapa	
Modificar etapa	
Listar etapas	

Tabla 27: CRC Etapa

TARJETA CRC	
Clase: Tema	
Descripción: Gestiona la información de los temas del sistema	
Atributos:	
Nombre	**Descripción**
ID	Identificador único del tema
Descripción	Descripción del tema
Responsabilidad:	**Colaboración:**
Crear tema	Habilidad
Eliminar tema	
Modificar tema	
Listar temas	

Tabla 28: CRC Tema

TARJETA CRC	
Clase: Registro	
Descripción: Gestiona la información del estudiante con sus evaluaciones y desempeño	
Atributos:	
Nombre	**Descripción**
ID	Identificador único de estudiante-datos
Nota-tcp	Nota del TCP del estudiante
Nota-seminario	Nota del seminario del estudiante
Promedio-pe	Promedio de preguntas escritas del estudiante
Promedio-po	Promedio de preguntas orales del estudiante
Inasistencia	Inasistencia del estudiante en H/C
Corte-evaluativo	Corte evaluativo del estudiante
Etapa-id	Identificador único de la etapa
Estudiante-usuario	Usuario único del estudiante

Responsabilidad:	Colaboración:
Crear estudiante-datos	Estudiante, Estudiante-Habilidad, Etapa
Eliminar estudiante-datos	
Modificar estudiante-datos	
Listar estudiante-datos	

Tabla 29: CRC Registro

Anexo 3 Tareas de programación

TAREAS DE PROGRAMACIÓN	
Historia de usuario	**Tarea de programación**
UH1 Adicionar usuario	1. Creación del modelo Usuario. 2. Creación del controlador Usuario para adicionar. 3. Definición de la interfaz adicionar usuario. 4. Definición de la URL crear usuario.
UH2 Modificar usuario	1. Creación del controlador Usuario para modificar. 2. Definición de la interfaz modificar usuario. 3. Definición de la URL modificar usuario.
UH3 Eliminar usuario	1. Creación del controlador Usuario para eliminar. 2. Definición de la interfaz eliminar usuario. 3. Definición de la URL eliminar usuario.
UH4 Listar usuarios	1. Creación del controlador Usuario para listar. 2. Definición de la interfaz listar usuario. 3. Definición de la URL listar usuario.
UH5 Autenticar usuario	1. Creación del controlador Usuario para listar. 2. Definición de la interfaz autenticar usuario. 3. Definición de la URL autenticar usuario.
UH6 Modificar contraseña	1. Creación del controlador Contraseña para modificar. 2. Definición de la interfaz modificar contraseña.

	3. Definición de la URL modificar contraseña.
UH7 Adicionar caso	1. Creación del modelo Caso. 2. Creación del controlador Caso para adicionar. 3. Definición de la interfaz adicionar caso. 4. Definición de la URL adicionar caso.
UH8 Modificar caso	1. Creación del controlador Caso para adicionar. 2. Definición de la interfaz adicionar caso. 3. Definición de la URL adicionar caso.
UH9 Eliminar caso	1. Creación del controlador Caso para eliminar. 2. Definición de la interfaz eliminar caso. 3. Definición de la URL eliminar caso.
UH10 Listar caso	1. Creación del controlador Caso para listar. 2. Definición de la interfaz listar caso. 3. Definición de la URL listar caso.
UH11 Adicionar estudiante	1. Creación del modelo Estudiante. 2. Creación del controlador estudiante para adicionar. 3. Definición de la interfaz adicionar estudiante. 4. Definición de la URL adicionar estudiante.
UH12 Modificar estudiante	1. Creación del controlador estudiante para modificar. 2. Definición de la interfaz modificar estudiante. 3. Definición de la URL modificar estudiante.
UH13 Eliminar estudiante	1. Creación del controlador estudiante para eliminar. 2. Definición de la interfaz eliminar estudiante. 3. Definición de la URL eliminar estudiante.

UH14 Listar estudiante	1. Creación del controlador estudiante para listar. 2. Definición de la interfaz listar estudiante. 3. Definición de la URL listar estudiante.
UH15 Adicionar habilidad	1. Creación del modelo Habilidades. 2. Creación del controlador habilidad para adicionar. 3. Definición de la interfaz adicionar habilidad. 4. Definición de la URL adicionar habilidad.
UH16 Modificar habilidad	1. Creación del controlador habilidad para modificar. 2. Definición de la interfaz modificar habilidad. 3. Definición de la URL modificar habilidad.
UH17 Eliminar habilidad	1. Creación del controlador habilidad para eliminar. 2. Definición de la interfaz eliminar habilidad. 3. Definición de la URL eliminar habilidad.
UH18 Listar habilidad	1. Creación del controlador habilidad para listar. 2. Definición de la interfaz listar habilidad. 3. Definición de la URL listar habilidad.
UH23 Adicionar remedial	1. Creación del modelo Remedial. 2. Creación del controlador remedial para adicionar. 3. Definición de la interfaz adicionar remedial. 4. Definición de la URL adicionar remedial.
UH24 Modificar remedial	1. Creación del controlador remedial para modificar. 2. Definición de la interfaz modificar remedial.

	3. Definición de la URL modificar remedial.
UH25 Eliminar remediales	1. Creación del controlador remedial para eliminar. 2. Definición de la interfaz eliminar remedial. 3. Definición de la URL eliminar remedial.
UH26 Listar remediales	1. Creación del controlador remedial para listar. 2. Definición de la interfaz listar remedial. 3. Definición de la URL listar remedial.
UH27 Proponer remedial	1. Implementación de los métodos para el agrupamiento. 2. Implementación de la función de semejanza entre casos 3. Implementación de la función de comparación entre rasgos.
UH31 Filtrar estudiante	1. Permite buscar a un estudiante determinado mediante el filtrando de cada uno de sus datos.

Tabla 30: Tareas de programación por historias de usuarios.

TAREA DE PROGRAMACIÓN	
No. de tarea: 2	**No.de HU: 1**
Nombre de la tarea: Creación del controlador Usuario para adicionar.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio:10-2-2016	**Fecha fin:**12-2-2016
Descripción: Se define la clase controladora encargada de realizar la lógica del negocio referente a los usuarios para realizar la acción adicionar.	

Tabla 31: TP2 Creación del controlador usuario para adicionar

TAREA DE PROGRAMACIÓN	
No. de tarea: 3	**No.de HU: 1**
Nombre de la tarea: Definición de la interfaz adicionar usuario	
Tipo de tarea: Desarrollo	**Responsable:** Ofelia Peláez Soto
Fecha inicio:13-2-2016	**Fecha fin:**15-2-2016

Descripción: Se define la interfaz visual para capturar la información requerida mediante la interacción de los usuarios con el sistema para realizar la acción adicionar.

Tabla 32: TP3 Interfaz usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 4	**No.de HU: 1**
Nombre de la tarea: Definición de la URL crear usuario.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 12-2-2016	**Fecha fin:**14-2-2016
Descripción: Se define la URL correspondiente a la acción adicionar usuario	

Tabla 33: TP4: Definición de la URL crear usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 1	**No.de HU: 1**
Nombre de la tarea: Creación del modelo Usuario	
Tipo de tarea: Desarrollo	**Responsable:** Ofelia Peláez Soto
Fecha inicio: 15-2-16	**Fecha fin:** 17-2-16
Descripción: Se crea el modelo encargado de persistir la información referente al usuario en la base de datos.	

Tabla 34: TP1 Creación del modelo Usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 2	**No.de HU: 1**
Nombre de la tarea: Creación del controlador Usuario para adicionar.	
Tipo de tarea: Desarrollo	**Responsable:** Ofelia Peláez Soto
Fecha inicio: 18-2-16	**Fecha fin:** 20-2-18
Descripción: Se define la clase controladora encargada de realizar la lógica del negocio referente a los usuarios para realizar la acción adicionar.	

Tabla 35: TP2 Creación del controlador Usuario para adicionar.

TAREA DE PROGRAMACIÓN	
No. de tarea: 3	**No.de HU: 1**
Nombre de la tarea: Implementación de la interfaz adicionar usuario	
Tipo de tarea: Desarrollo	**Responsable:** Ofelia Peláez Soto
Fecha inicio: 21-2-16	**Fecha fin:** 23-2-16
Descripción: Se define la interfaz visual para capturar la información requerida mediante la interacción de los usuarios con el sistema para realizar la acción adicionar.	

Tabla 36: TP3 Implementación de la interfaz adicionar usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 4	**No.de HU: 1**
Nombre de la tarea: Definición de la URL crear usuario.	
Tipo de tarea: Desarrollo	**Responsable:** Ofelia Peláez Soto
Fecha inicio: 24-2-16	**Fecha fin:** 26-2-16

Descripción: Se define la URL correspondiente a la acción adicionar usuario

Tabla 37: TP Definición de la URL crear usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 1	**No.de HU: 2**
Nombre de la tarea: Creación del controlador Usuario para modificar.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 28-2-16	**Fecha fin:** 1-3-16
Descripción: Se define la clase controladora encargada de realizar la lógica del negocio referente a los usuarios para realizar la acción modificar.	

Tabla 38: TP1 Creación del controlador Usuario para modificar

TAREA DE PROGRAMACIÓN	
No. de tarea: 2	**No.de HU: 2**
Nombre de la tarea: Implementación de la interfaz modificar usuario	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 2-3-16	**Fecha fin:** 4-3-16
Descripción: Se define la interfaz visual para capturar la información requerida mediante la interacción de los usuarios con el sistema en correspondencia con las modificaciones que se realicen.	

Tabla 39: TP2 Implementación de la interfaz modificar usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 3	**No.de HU: 2**
Nombre de la tarea: Definición de la URL modificar usuario.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 5-3-16	**Fecha fin:** 7-3-16
Descripción: Se define la URL correspondiente a la acción modificar usuario	

Tabla 40: TP3 Definición de la URL modificar usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 1	**No.de HU:** 3
Nombre de la tarea: Creación del controlador Usuario para eliminar.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 8-3-16	**Fecha fin:** 10-3-16
Descripción: Se define la clase controladora encargada de realizar la lógica del negocio referente a los usuarios para realizar la acción eliminar.	

Tabla 41: TP1 Creación del controlador Usuario para eliminar

TAREA DE PROGRAMACIÓN	
No. de tarea: 2	**No.de HU: 3**
Nombre de la tarea: Definición de la interfaz eliminar usuario	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 11-3-16	**Fecha fin:** 13-3-16

Descripción: Se define la interfaz visual para capturar la información requerida mediante la interacción de los usuarios con el sistema para realizar la acción eliminar.

Tabla 42: TP2 Definición de la interfaz eliminar usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 3	**No.de HU:** 3
Nombre de la tarea: Definición de la URL eliminar usuario.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 14-3-16	**Fecha fin:** 16-3-16
Descripción: Se define la URL correspondiente a la acción eliminar usuario	

Tabla 43: Definición de la URL eliminar usuario.

TAREA DE PROGRAMACIÓN	
No. de tarea: 1	**No.de HU: 4**
Nombre de la tarea: Creación del controlador Usuario para listar.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 17-3-16	**Fecha fin:** 19-3-16
Descripción: Se define la clase controladora encargada de realizar la lógica del negocio referente a los usuarios para realizar la acción listar.	

Tabla 44: Creación del controlador Usuario para listar.

TAREA DE PROGRAMACIÓN	
No. de tarea: 2	**No.de HU: 3**
Nombre de la tarea: Implementación de la interfaz listar usuario	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 20-3-16	**Fecha fin:** 22-3-16
Descripción: Se define la interfaz visual que capturará la información requerida mediante la interacción de los usuarios con el sistema para realizar la acción listar, en la misma se visualizarán todos los usuarios del sistema con las acciones modificar, eliminar y ver.	

Tabla 45: Implementación de la interfaz listar usuario

TAREA DE PROGRAMACIÓN	
No. de tarea: 3	**No.de HU:** 4
Nombre de la tarea: Definición de la URL listar usuario.	
Tipo de tarea: Desarrollo	**Responsable:** Yeinelis Hierrezuelo Ramírez
Fecha inicio: 23-3-16	**Fecha fin:** 25-3-16
Descripción: Se define la URL con el nombre *list_user* correspondiente a la acción listar usuario, vinculando la interfaz listar usuario con el controlador.	

Tabla 46: Definición de la URL listar usuario.

Anexo 4: Pruebas de aceptación

PRUEBA DE ACEPTACIÓN	
Código: UH11_P2	**HU:** 11
Nombre: Adicionar estudiante	

Descripción: El administrador del sistema una vez accedido al mismo luego de ser autenticado, seleccionará del menú la opción “Nuevo estudiante”. Se le mostrará una ventana con los datos requeridos para realizar la operación, luego de ser insertados los mismos, el sistema realizará su validación. De no ser correctos el sistema debe mostrar un mensaje de error y no realizar la operación de inserción.
Condiciones de ejecución: Usuario autenticado con permisos de administración
Pasos de ejecución: 1. El administrador se autentica en el sistema. 2. Del menú principal seleccionará “Estudiante”- “Nuevo estudiante”. 3. Se le mostrará una interfaz solicitando los parámetros: “Nombre y apellidos” y “Usuario” 4. El administrador introduce parámetros incorrectos y presiona el botón “Aceptar”. 5. El sistema verifica los parámetros. 6. El sistema muestra un mensaje de error.
Resultados esperados: El sistema mostrará un mensaje de error y no realizará la inserción del remedial en la base de datos.
Resultado obtenido: El sistema mostró un mensaje de error pero realizó la inserción del estudiante en la base de datos.
Evaluación de la prueba: Insatisfactoria

Tabla 47: PA Adicionar estudiante

PRUEBA DE ACEPTACIÓN	
Código: UH1_P1	**HU:** 1
Nombre: Adicionar usuario	
Descripción : El administrador del sistema una vez entrado en el sistema seleccionará del menú la opción “Crear usuario”. Se le mostrará un formulario donde insertará los parámetros: Nombre y apellidos y usuario, internamente el sistema validará los datos. De ser correctos mostrará la lista de usuarios del sistema y se introducirán los datos en la base de datos del sistema.	
Condiciones de ejecución : Administrador autenticado en el sistema	
Pasos de ejecución: 1. El administrador se autentica en el sistema. 2. Del menú principal seleccionará “Usuarios”- “Crear usuario”. 3. Se le mostrará una pantalla solicitando el Nombre y apellidos, y Usuario. 4. El administrador introduce ambos y presiona el botón “Aceptar”. 5. El sistema válida los campos y muestra el formulario para el “Modificar usuario”.	
Resultados esperados: Se insertará el usuario, visualizándose en la lista de usuarios del sistema.	
Resultado obtenido: Se insertó el usuario en la base de datos y se visualizó en la lista de usuarios del sistema.	

Evaluación de la prueba: Satisfactoria

Tabla 48: PA Adicionar usuario

Anexo 5: Pruebas unitarias

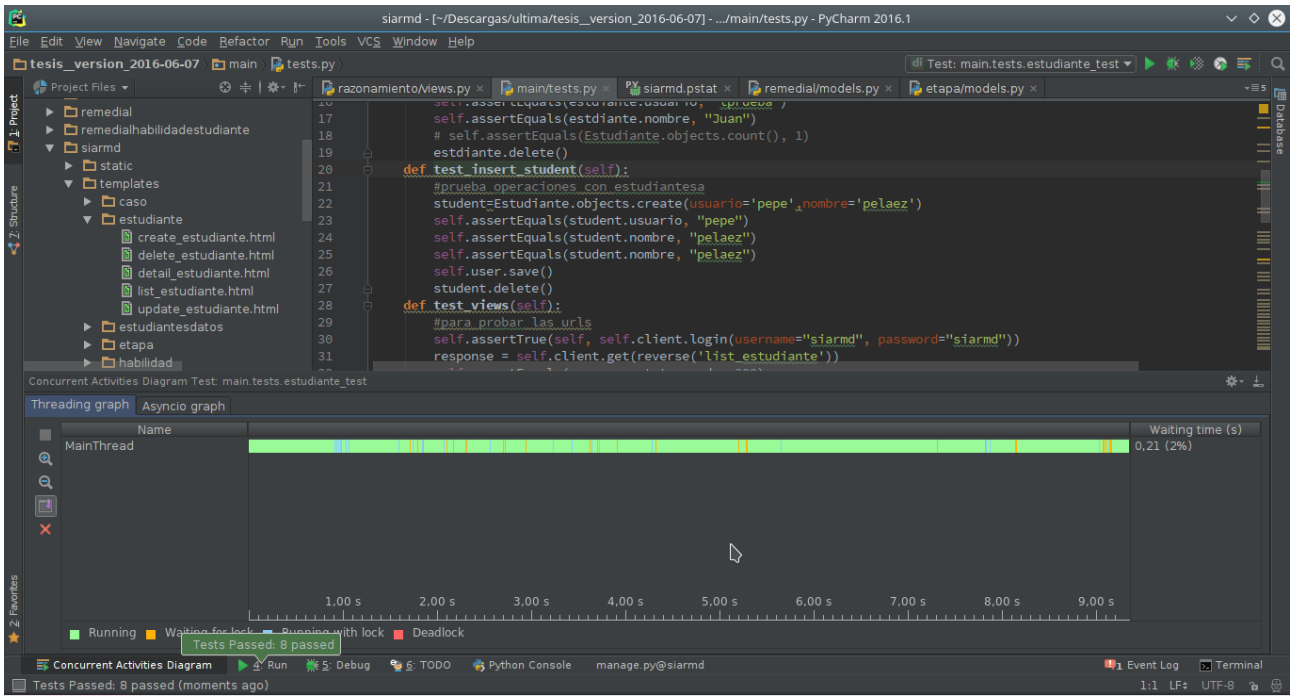

Figura 5: Diagrama de ejecución de pruebas.

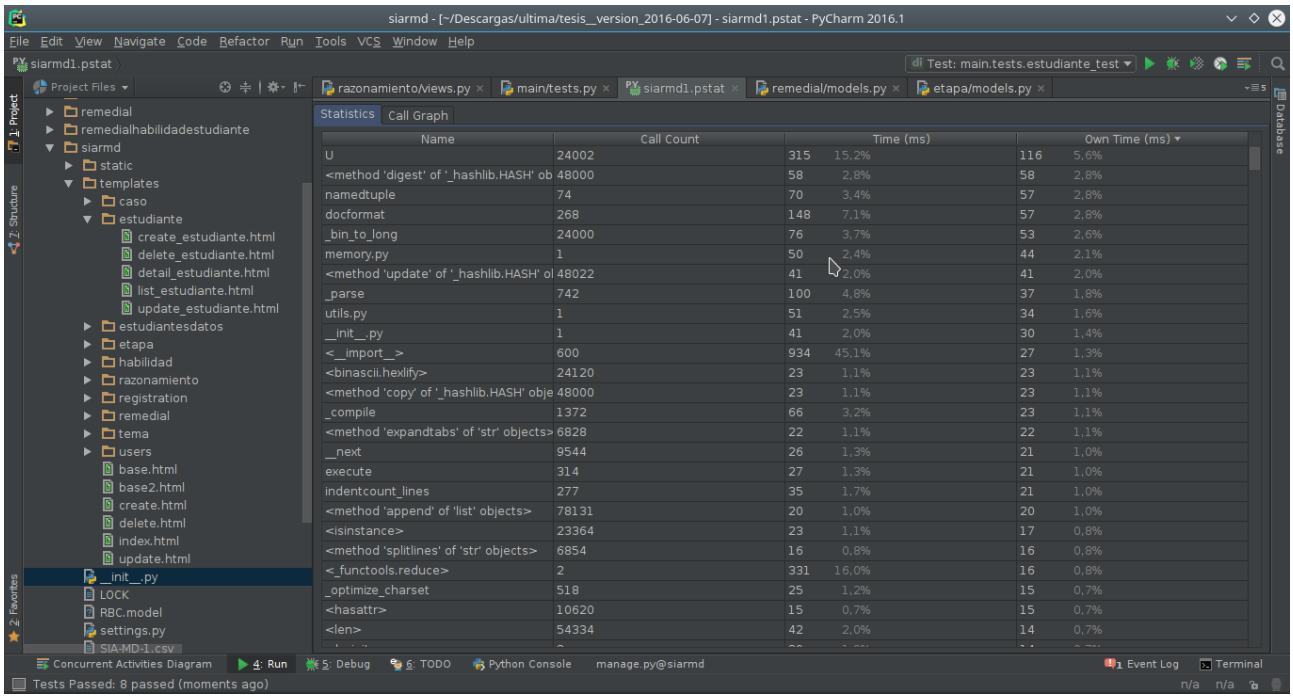

Figura 6: Estadísticas de ejecución de las pruebas.

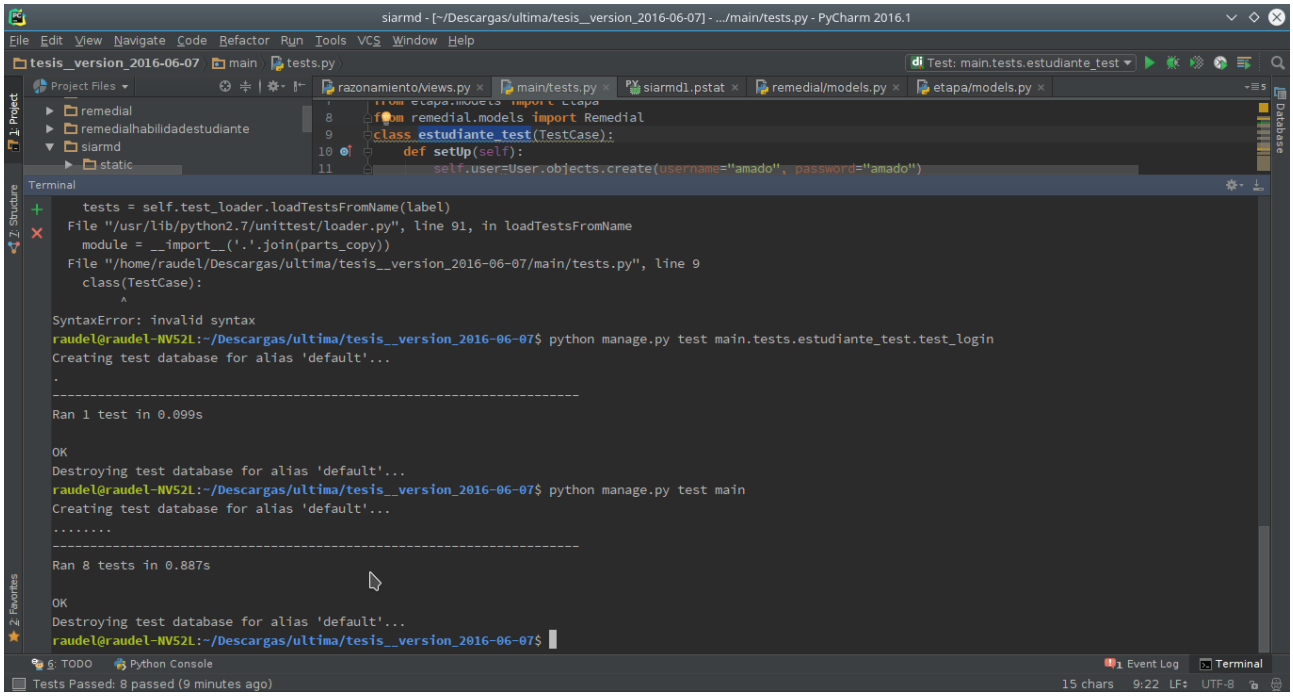

Figura 7: Resultados de la ejecución de las pruebas.

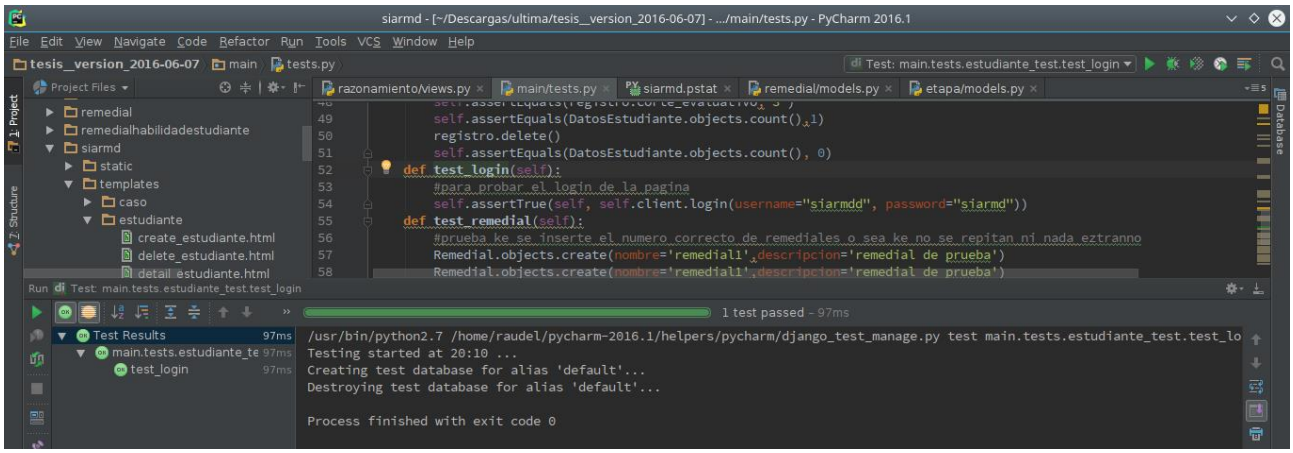

Figura 8: Prueba a la autenticación del sistema.

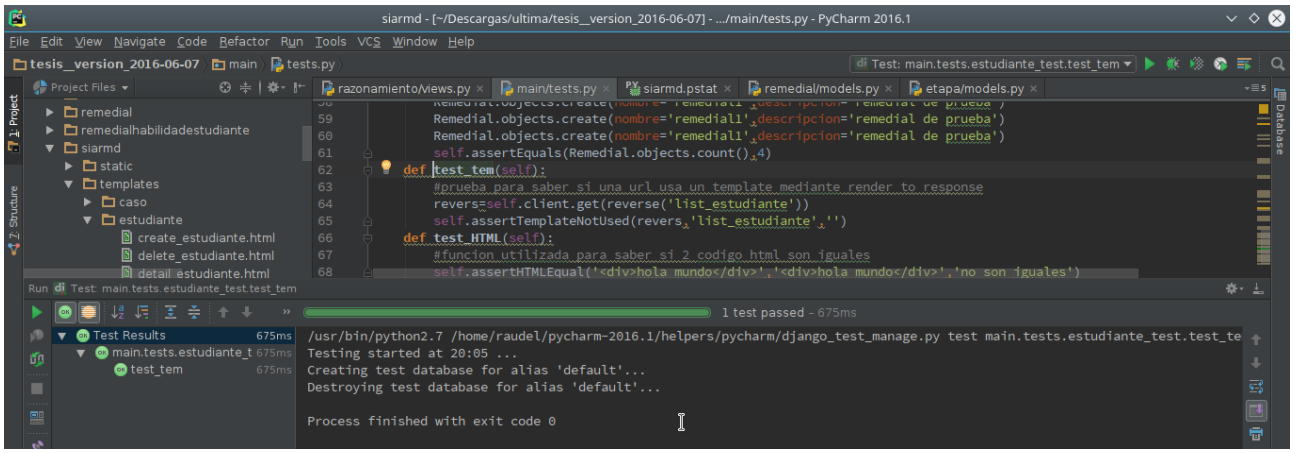

Figura 9: Prueba para determinar si una url usa la template correspondiente mediate render to response.

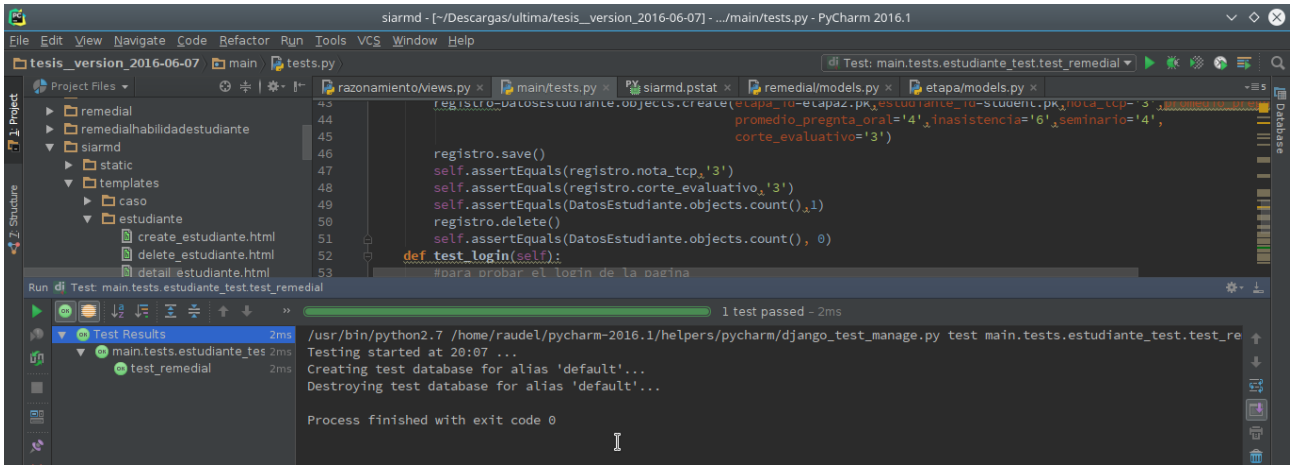

Figura 10: Prueba para las acciones eliminar, insertar, modificar, listar e inspeccionar del módulo Remedial