UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

Título: Sistema de Enseñanza-Aprendizaje Inteligente de apoyo al curso introductorio de Matemática.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

Autor(es): Lilidiana Cruz Pazos.
José Enrique Pacheco Piñeyro.

Tutor(es): Dra. Natalia Martínez Sánchez.

Ciudad de la Habana, junio 23 del 2010.
Año 52 de la Revolución.

A mi madre, por la fuerza y el amor con que me ha dirigido por la vida.
A mi padre, por todo el sacrificio de los últimos años, por el amor que sé que hay en él para mí.
A Danner Marante, por estar siempre cuando lo he necesitado, porque aún es mi confesionario, la persona que me conoce más que nadie en este mundo.
Lilidiana.

Le dedico el presente trabajo a mi madre, por ser madre, padre, por ser todo en mi vida.
A mis abuelos Silvia, Toti y Cuca.
A mis tíos, hermanos y primos.
José Enrique.

AGRADECIMIENTOS

Sabiendo que jamás encontraré la forma de agradecer el constante apoyo y confianza de las personas que contribuyeron a que terminara satisfactoriamente esta etapa de mi vida, sólo espero que comprendan que mis ideales, esfuerzos y logros han sido también suyos e inspirados en ustedes.

A mis padres, por el apoyo y la confianza de siempre.
A Danier, por las maravillas que hace con mi autoestima.
A mi Tía Teresa, por ser como mi madre estos cinco años.
A Alejandro, porque gracias a él, mi vida tomó la dirección que me llevó hasta aquí.
A Katia que ha sido una gran amiga aún cuando no ha aprobado mis acciones.
A Pedro por toda la ayuda brindada, por convertirse en el amigo que nunca imaginé.
A Dayron por hacer de cada momento un rato feliz, por toda la dedicación y por cambiar mi mala actitud ante la vida.
A Inés y Alber por abrirme las puertas que tanto necesito.
A Beni, Eduardo, Chino, Adriana, Doralqui, Leivis, Yasmanis, Yunior, Melanie, Leonardo y Dayisel, por ser mi familia de verdad entre tantos que no podrán unirse nunca.
A Yander, por hacer tanto cuando la posibilidad de graduarme este curso era casi nula.
A Blanca Elena por estar ahí par mi madre estos años y por la ayuda indirecta para mí
A todos los amigos que no voy a mencionar pero que están siempre en mi corazón.

Lilidiana.

Quiero agradecer a Dios.
A mi madre por guiarme, aconsejarme y haberme formado como la persona que soy.
A mis abuelos por todo el apoyo que me han dado siempre.
A mi tío Fernando y mi hermano Pacheco por ser el ejemplo de persona que quiero ser.
A mi tía Marilin.
A todos mis amigos y a todo aquel que se preocupó por mi

José Enrique.

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 9 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2010.

AUTOR
Lilidiana Cruz Pazos

AUTOR
José Enrique Pacheco Piñeyro

TUTOR
Dra. Natalia Martínez Sánchez

La presente investigación se realizó en la Universidad de las Ciencias Informáticas (UCI) ubicada en Ciudad de la Habana, Cuba, durante el período comprendido entre noviembre del año 2009 y junio del 2010. En la misma se vieron implicados algunos profesores del Departamento de Ciencias Básicas de la Facultad 9, además de los diplomantes que la llevaron a cabo. El propósito de esta investigación consistió en el desarrollo de un Sistema de Enseñanza-Aprendizaje Inteligente (SEAI) de apoyo al proceso de enseñanza-aprendizaje de la matemática en el curso introductorio que se imparte en la UCI. Para ello fue necesaria: la evaluación y el análisis del estado del arte en la ayuda a la toma de decisiones, el control y el seguimiento de estudiantes en el proceso de enseñanza-aprendizaje de matemática; el desarrollo de un modelo basado en herramientas inteligentes, que posibiliten la ayuda en la toma de decisiones en el diagnóstico de estudiantes en el proceso de enseñanza-aprendizaje de matemática. El trabajo investigativo se aborda desde la perspectiva de los tipos de estudio empíricos, con la aplicación de un diseño de investigación de campo donde la población estuvo formada por trescientos cuatro (304) estudiantes, de los cuales se tomó una muestra del 60% que proporciona un tamaño muestral de ciento ochenta y dos (182) estudiantes. El resultado final se revierte en que el estado del arte en la ayuda a la toma de decisiones, el control y el seguimiento de estudiantes en el proceso de enseñanza-aprendizaje de matemática en la UCI, se ha visto ampliado a partir de la existencia de una nueva herramienta de software con funcionalidades que responden como solución a los problemas que supone este campo.

PALABRAS CLAVES

Sistemas de enseñanza aprendizaje inteligente, Razonamiento basado en casos.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

El aprendizaje es una etapa importante en el desarrollo de toda actividad educativa, por ello es necesario exigir para la enseñanza las más calificadas ayudas tecnológicas (Carbonell, 1970).

Dentro de esta línea se presenta el desafío del desarrollo de sistemas tutoriales con características tales que abarquen el tratamiento inteligente de información relacionada con un cierto dominio, el modelado del conocimiento del estudiante, disímiles estrategias de enseñanza y una interfaz para la comunicación con el estudiante (Carbonell, 1970). Los investigadores en el área han mirado con esperanza los adelantos de las investigaciones técnicas y herramientas de la Inteligencia Artificial (IA), la Pedagogía y la Psicología Cognoscitiva para lograr los objetivos planteados.

Las características propias de la Enseñanza Asistida por Computadoras (EAC) en lo que se refiere a la consideración de los rasgos individuales del estudiante, el diagnóstico de las causas de sus errores y al tratamiento de los mismos en el proceso de enseñanza-aprendizaje han condicionado a los lenguajes y sistemas de autor (programas informáticos que facilitan la creación de productos multimedia a usuarios sin conocimientos de programación (Zazpe, 1998) a evolucionar en conexión con los avances sobre los Sistemas de Enseñanza-Aprendizaje Inteligentes (SEAI) denominados también Sistemas Tutoriales Inteligentes (STI).

En la Universidad de las Ciencias Informáticas (UCI) donde se forman profesionales de la industria de software de Cuba, se hace necesaria la utilización de alternativas y técnicas novedosas que aseguren en cierto grado la enseñanza de las materias básicas y avanzadas de la carrera a los estudiantes. De modo que una posible solución para ampliar la gama de herramientas disponibles por la comunidad universitaria para su autoestudio son los SEAI. El hecho de que sean herramientas de última generación no necesariamente asegura que sea la mejor propuesta pero al menos es: una propuesta. Es necesario aclarar que el grado de aseguramiento de la enseñanza del que se habla anteriormente viene dado en la medida que los sistemas puedan registrar y reportar los datos distintivos del desempeño de cada estudiante y los profesores sean capaces de comprenderlos y actuar en consecuencia.

En función de utilizar el conocimiento almacenado en los datos de los registros de notas de los diagnósticos y pruebas finales de matemática de cursos anteriores se ha identificado como **problema científico** que no contar con herramientas que apoyen el proceso de enseñanza-aprendizaje personalizado de los estudiantes en el curso introductorio de Matemática de la Universidad de las Ciencias Informáticas puede influir negativamente en los resultados de estos estudiantes en el curso. El **objeto de estudio** de esta investigación lo constituye el proceso de evaluación y atención diferenciada a los estudiantes en el curso introductorio de Matemática, siendo

el **campo de acción** el desarrollo de un software educativo adaptativo en los temas que se imparten en el curso introductorio de Matemática. Por tal razón se plantea como **idea a defender**: El desarrollo de una herramienta de apoyo al proceso de enseñanza-aprendizaje en el curso introductorio de Matemática que realice los procesos de diagnóstico, tratamiento y seguimiento de los estudiantes influye de forma positiva en el proceso de enseñanza aprendizaje de los estudiantes.

Se definió como **objetivo general** de la investigación desarrollar un SEAI de apoyo al proceso de enseñanza-aprendizaje de la matemática en el curso introductorio que se imparte en la UCI. Los **objetivos específicos** son:

- Evaluar y analizar el estado del arte en la ayuda a la toma de decisiones, el control y el seguimiento de estudiantes en el proceso de enseñanza-aprendizaje de matemática en el curso introductorio.
- Desarrollar un modelo basado en técnicas de Inteligencia Artificial, que faciliten la toma de decisiones en el diagnóstico de estudiantes en el proceso de enseñanza-aprendizaje de matemática.

  Validar los resultados obtenidos en el sistema que se desarrolle.

Es necesario especificar que el alcance de este último objetivo se reduce a la comprobación del correcto funcionamiento del producto para agrupar y brindar el material que corresponda a cada estudiante y no a lo referente a si este aprende o no.

Las **tareas** a desarrollar durante la investigación son:

- Describir el proceso de evaluación y atención diferenciada a los estudiantes en el curso introductorio de Matemática.
- Describir el estado actual de los STI y las técnicas de IA más utilizadas en el desarrollo de los mismos.
- Desarrollar un software educativo adaptativo a los conocimientos de los estudiantes sobre la asignatura Matemática.
- Evaluar solución encontrada.

Para el desarrollo de las tareas científicas se han combinado diferentes **métodos** y procedimientos teóricos y empíricos de la investigación científica en la búsqueda y procesamiento de la información. Los fundamentales son:

Métodos Teóricos:

El método **analítico-sintético** predomina en la etapa de estudio de los factores que condicionan el proceso de aprendizaje de los estudiantes en el curso de Matemática I. El análisis revela los diferentes factores que inciden en este proceso como por ejemplo:

- Las condiciones de estudio de los estudiantes en la UCI y su nivel de desarrollo intelectual.

- Frecuencia de tiempo y recursos dedicados al autoestudio.
- Atención diferenciada que recibe el estudiante sobre el éxito o fracaso de su proceso de aprendizaje y el nivel de información que se le brinda.

Mientras que la síntesis permite descubrir las relaciones existentes entre los factores identificados, y el condicionamiento mutuo que ejercen en el proceso de aprendizaje; el **método inductivo-deductivo** como vía de la constatación teórica durante el desarrollo de la tesis; el **método histórico-lógico** se manifiesta el estudio crítico de los trabajos anteriores y en la revisión de los registros de notas de los estudiantes tomados como muestra, así como las caracterizaciones cualitativas de la Matemática I en el primer semestre de los cursos 2007-2008 y 2008-2009. Esto se evidencia de manera más profunda en el epígrafe 1.3; el **método sistémico** en el desarrollo del sistema computacional y lograr que los elementos que formen parte de la aplicación real sean un todo que funcione de manera armónica; el **método de modelación** en el desarrollo de los algoritmos y diagramas presentados.

Métodos matemáticos:

Métodos estadísticos para la validación de la necesidad de desarrollar un Sistema de Enseñanza-Aprendizaje inteligente como herramienta para el apoyo al estudio independiente en el curso introductorio de matemática en la UCI.

Métodos Empíricos:

El **método experimental** en la comprobación de la utilidad de los resultados obtenidos; la **entrevista** realizada a algunos de los profesores de Matemática del Departamento de Ciencias Básicas de la Facultad 9.

El siguiente trabajo está estructurado en tres capítulos principales.

Capítulo 1. Fundamentación teórica. En este capítulo se analizan elementos teóricos tales como: Descripción del objeto de estudio, sistemas similares existentes vinculados a las Técnicas de Razonamiento Basado en Casos aplicables a Sistemas de Enseñanza-Aprendizaje.

Capítulo 2. Tendencias y Tecnologías a utilizar. En este capítulo se fundamenta el uso de las herramientas y metodologías necesarias para el desarrollo de la aplicación propuesta.

Capítulo 3. Presentación de la solución propuesta. Este capítulo incluye el modelo de dominio, la especificación de requisitos, la descripción de los casos de uso del sistema y la descripción de la arquitectura a utilizar.

Capítulo 4. Construcción de la solución propuesta. Este capítulo incluye todo lo referente a la construcción del sistema desde el diseño hasta la implementación de la aplicación.

Capítulo 5. Pruebas del sistema desarrollado. En este capítulo se muestran los casos de prueba realizados al sistema a partir de la aplicación de las técnicas de pruebas de caja negra.

CAPÍTULO 1

Fundamentación teórica.

1.1 Introducción

En este capítulo se conceptualizan los Sistemas de Enseñanza-Aprendizaje Inteligentes. Se analiza cómo el proceso del modelado del alumno ha motivado la aplicación de técnicas de Inteligencia Artificial. Se comenta acerca de SEAI y herramientas de autor para el desarrollo de Sistema de Enseñanza-Aprendizaje que existen en la actualidad. Finalmente se enfatiza y se describe el Razonamiento Basado en Casos como una alternativa para la elaboración de los SEAI, valorándose las ventajas de éste.

1.2 Conceptos asociados al dominio del problema

Los SEAI[1] son programas que portan conocimientos sobre cierta materia y cuyo propósito es trasmitir este conocimiento a los alumnos mediante un proceso interactivo individualizado, intentando simular la forma en que un tutor o profesor guiaría al alumno en el proceso de enseñanza-aprendizaje (Alpigini, 2002), (Salgueiro, 2005), (Shneiderman, 2006), (Sierra, 2006).

El término inteligente se refiere a la habilidad del sistema sobre qué enseñar, cuándo enseñar y cómo enseñar, simulando la actividad de un profesor real. Para lograrlo, un SEAI debe encontrar la información relevante sobre el proceso de aprendizaje de ese estudiante y aplicar el mejor medio de instrucción según sus necesidades individuales (Huapaya, 2005), (Ovalle, 2005), (Trella, 2006), (Jiménez y Ovalle, 2008), (Gómez, 2008), (Cataldi y Lage, 2009)

Los trabajos de Barr, 1982; Tsinakos, 2000; Millán, 2000; Franco, 2001; Urretavizcaya, 2001; Millán, 2000; Conejo, 2001; Ferrero, 2001; Romero, 2002; González, 2004; Parra, 2004; Iglesias, 2004; Salgueiro, 2005; Cataldi, 2006; Sánchez y Lama, 2007; Viccari, 2007; Arias, 2007; Ovalle, 2007; Huapaya, 2007; Riccucci, 2008, entre otros, consultados en la literatura, describen la arquitectura básica de los SEAI, y sólo difieren su definición en el número de módulos y como se nombran los mismos, existiendo pleno consenso en que el aspecto distintivo de un SEAI es el Modelo del Estudiante.

[1] En el trabajo se utilizará SEAI tanto para el plural como para el singular.

La arquitectura descrita en Ovalle, 2007, mostrada en la figura 1 reúne los elementos más comúnmente encontrados en la literatura consultada y se resumen en el criterio que plantea que un SEAI está compuesto por un módulo del dominio, un módulo del alumno y el módulo pedagógico, que operan de forma interactiva y se comunican a través de un módulo central que suele denominarse módulo entorno.

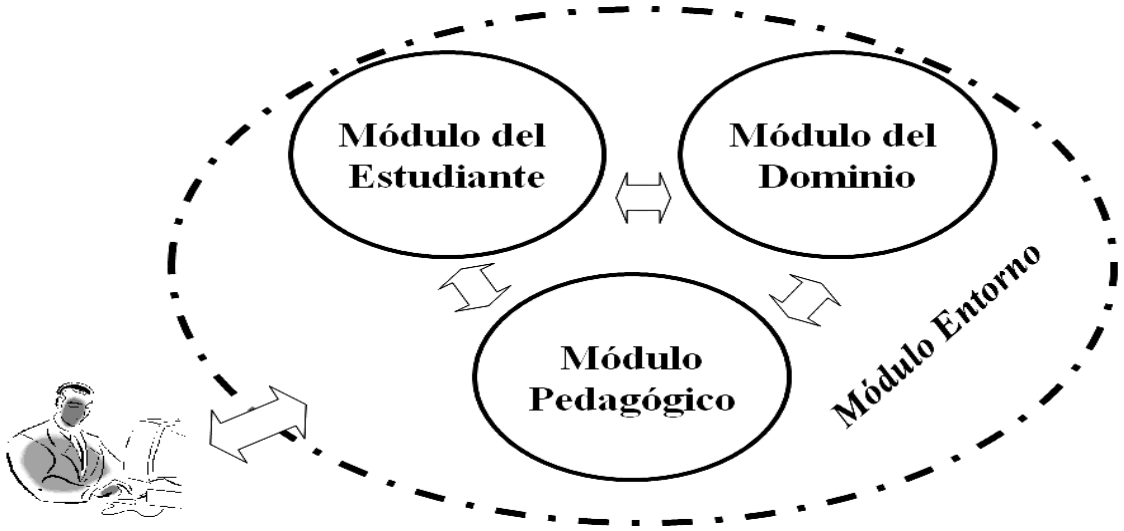


Figura 1. Arquitectura de un Sistema de Enseñanza Aprendizaje Inteligente.

1.2.1 Módulo del Estudiante

El módulo del estudiante está presente en todos los trabajos en los que se describe la arquitectura básica de un SEAI. Generalmente solo se diferencian entre sí por las características a incluir para representar el modelo del estudiante.

Puede afirmarse que el modelo del estudiante es un problema de investigación que debe enfocarse desde todas sus aristas con el fin de obtener una representación de las características del estudiante completa y precisa. Algunos autores como se referencia a continuación toman en consideración características tales como: el estilo de aprendizaje, el nivel de conocimiento, la información personal o la combinación de algunas de ellas:

Estilos de aprendizaje: conjunto de características psicológicas, rasgos cognitivos, afectivos y fisiológicos que suelen expresarse conjuntamente cuando una persona debe enfrentar una situación de aprendizaje. Los rasgos cognitivos tienen que ver con la forma en que los estudiantes estructuran los contenidos, forman y utilizan conceptos, interpretan la información, resuelven los problemas, etc.

Los rasgos afectivos se vinculan con las motivaciones y expectativas que influyen en el aprendizaje, mientras que los rasgos fisiológicos están relacionados con el biotipo y el biorritmo del estudiante (Graf, 2005), (Durán, 2006), (Duque, 2007), (Carmona y Castillo, 2007).Nivel de

conocimiento: características propias de cada estudiante referente al grado de conocimiento que posee acerca de conceptos, temas y asignaturas (Barr, 1982), (Holt, 1994), (Costa, 2005), (Duque, 2007).

Información personal: datos como la edad, género, idioma, y otras informaciones que pueda ser de interés (Costa, 2005), (Durán, 2006), (Duque, 2007).

1.2.2 Módulo del Dominio

El módulo del dominio, denominado también por muchos autores como módulo experto, proporciona los conocimientos del dominio. Satisface dos propósitos diferentes. En primer lugar, presentar la materia de la forma adecuada para que el alumno adquiera las habilidades y conceptos, lo que incluye la capacidad de generar preguntas, explicaciones, respuestas y tareas para el alumno.

En segundo lugar, el módulo del dominio debe ser capaz de resolver los problemas generados, corregir las soluciones presentadas y aceptar aquellas soluciones válidas que han sido obtenidas por medios distintos.

En este módulo, el conocimiento a ser enseñado por el SEAI debe organizarse pedagógicamente para facilitar el proceso de enseñanza-aprendizaje, (Ming y Quek, 2007).

1.2.3 Módulo Pedagógico

Decide qué, cómo y cuándo enseñar los contenidos del tutor, adaptando sus decisiones pedagógicas a las necesidades del estudiante (Jiménez y Ovalle, 2008). Algunos autores le denominan módulo tutor, ya que es el encargado de comparar las características de los estudiantes con el contenido a enseñar y elegir la mejor forma de tomar las decisiones pedagógicas oportunas, adaptándose en cada momento al estudiante.

1.2.4 Módulo Entorno

El módulo entorno gestiona la interacción de las otras componentes del sistema y controla la interfaz persona-computadora. Especifica y da soporte a las actividades del estudiante y a los métodos que se usan para realizar dichas actividades. Los entornos deben ser fáciles de utilizar y atractivos, de forma que el alumno pierda el mínimo tiempo posible en aprender a utilizar el entorno y pueda centrar toda su atención en el proceso de enseñanza-aprendizaje del contenido.

La necesidad de crear ambientes computacionales capaces de mantener el interés de sus usuarios, implica el desarrollo de interfaces personalizadas para ofrecer un servicio que permita un trato individualizado al usuario en particular (Medina, 2007), adaptando su interacción con el sistema a sus necesidades e intereses personales (Paredes, 2001). Para ofrecer un servicio adecuado a cada usuario es necesario que el sistema cuente con una representación de sus características

propias, y en base a dicha representación, tome las decisiones acertadas en la interacción con el usuario (Fischer, 2000), (Costa, 2005).

1.3 Objeto de Estudio

Surge como consecuencia del problema de investigación o científico, por lo que no debe surgir antes. Respetar esta sugerencia es primordial para el éxito en la planificación de la investigación en función del problema detectado. Siempre se da en proceso y como tal, es tan amplio y complejo que no es recomendable para la excelencia de la investigación abordar todos sus componentes, por ello es muy importante definir de dicho proceso un campo de acción. La aplicación de los resultados de las investigaciones no depende de que el objeto sea abordado íntegramente o no, sino de la calidad con que realmente sea definido y resuelto el campo de acción. (Guardo, 2010)

1.3.1 Descripción General

Según los datos proporcionados por el Departamento de Ciencias Básicas de la Facultad 9 de la UCI, durante el curso 2007-2008 de los 150 estudiantes que matricularon en la facultad 9 en el nuevo ingreso, 93 de ellos (cifra equivalente al 62 porciento del total) obtuvieron en el diagnóstico inicial de Matemática los resultados que se resumen en la tabla 1 y que son representados en el gráfico 1 de manera más clara.

Tabla 1 Resultados Diagnóstico.2007-2008.

Puntuación	Cantidad de Estudiantes	Porciento del Total
5 puntos	**15**	**16**
4 puntos	**20**	**21**
3 puntos	**18**	**19**
Menos de 2 puntos	**40**	**43**

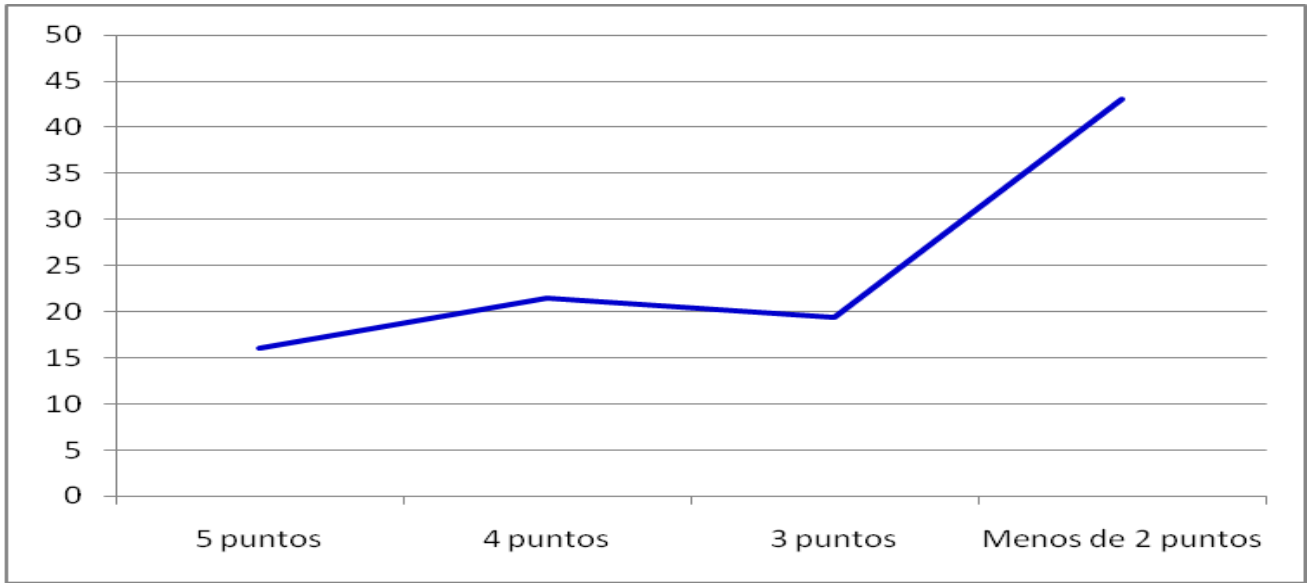


Figura 2. Resultados Diagnóstico.2007-2008.

Al terminar el semestre después de aplicar el plan de estudio G los resultados se comportaron del modo que se muestra en la tabla 2 y el gráfico 2.

Tabla 2 Resultados Prueba Final. 2007-2008

Puntuación	Cantidad de Estudiantes	Porciento del Total
5 puntos	**12**	**13**
4 puntos	**22**	**24**
3 puntos	**20**	**21**
Menos de 2 puntos	**39**	**42**

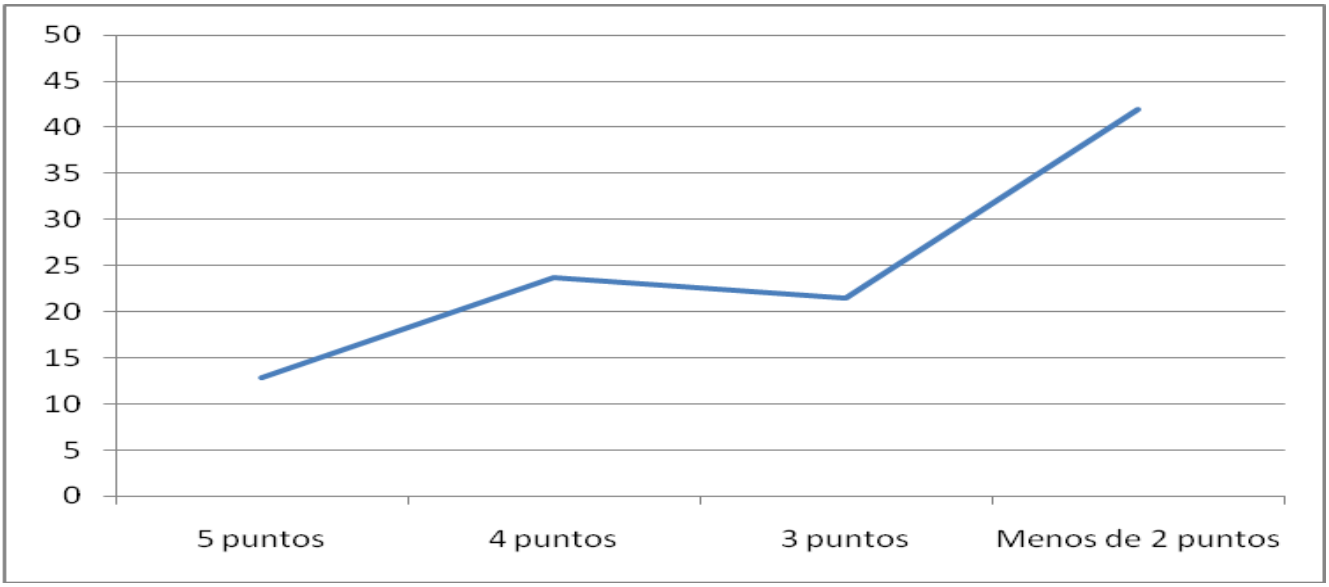


Figura 3. Resultados Prueba Final. 2007-2008.

Si se analizan ambos resultados a partir de la representación en el gráfico 3 se puede concluir que:

Durante el Curso 2007-2008 los estudiantes que obtuvieron resultados excelentes en el diagnóstico inicial disminuyeron su rendimiento en la prueba final del primer semestre. Los que obtuvieron cuatro puntos aumentaron en un 2 porciento en comparación con los que alcanzaron dicha nota al principio del curso. Con respecto a la calificación de 3 puntos, los resultados finales se comportan como una cota superior de los iniciales ya que los supera en un 1 porciento, de modo que hay una tendencia de aumento a la calificación de 3, situación que se agrava cuando se analizan los que obtuvieron 2 que sólo disminuyeron en un 1 porciento.

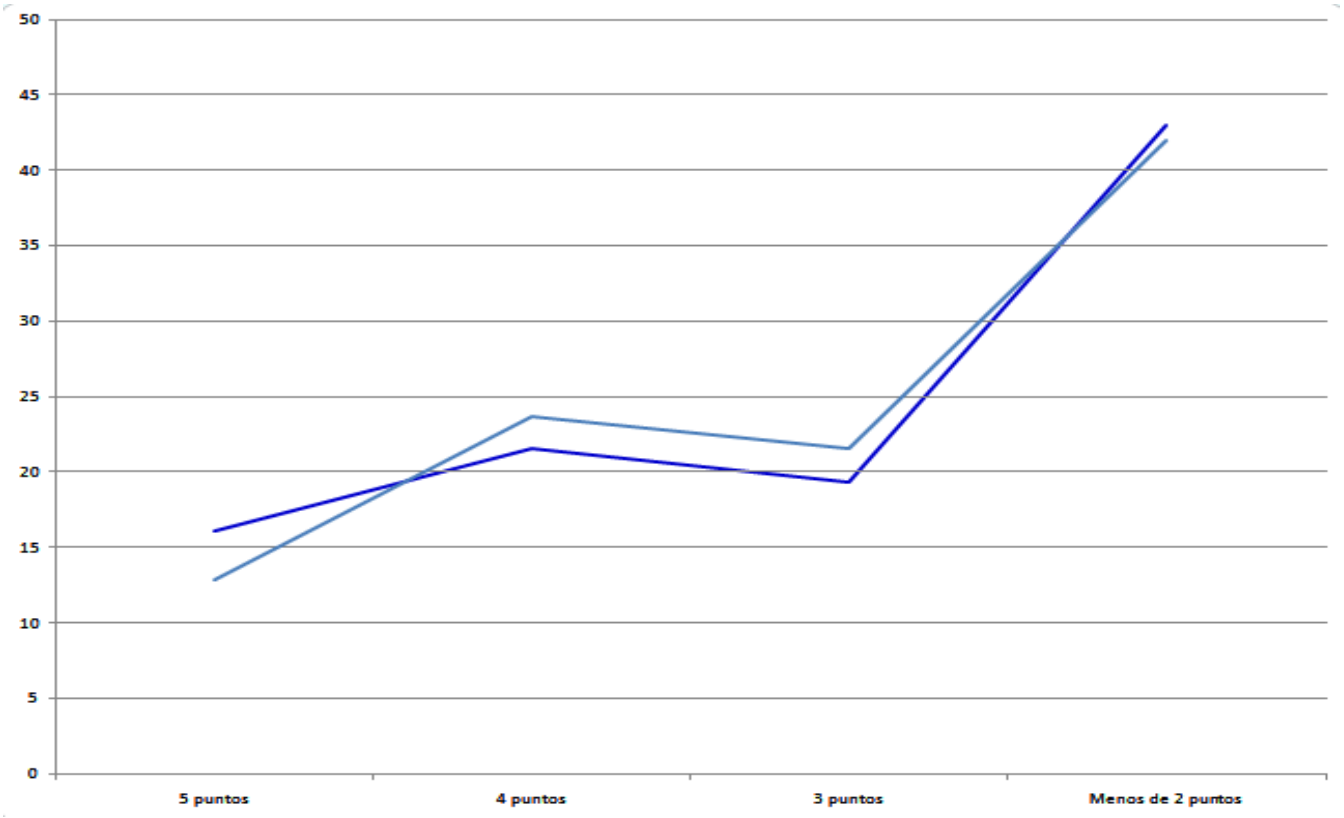


Figura 4. Superposición de resultados. 2007-2008.

En el transcurso del período lectivo de 2008-2009 los primeros cuatro grupos de nuevo ingreso contaban con 89 estudiantes de un total de 154 en el año. Esto representa un 58 porciento de todos los matriculados. Los resultados para la misma prueba esta ocasión fueron los que se muestran en la tabla 3 y el gráfico 4.

Tabla 3 Resultados Diagnóstico.2008-2009.

Puntuación	Cantidad de Estudiantes	Porciento del Total
5 puntos	**13**	**14**
4 puntos	**24**	**26**
3 puntos	**21**	**23**
Menos de 2 puntos	**31**	**34**

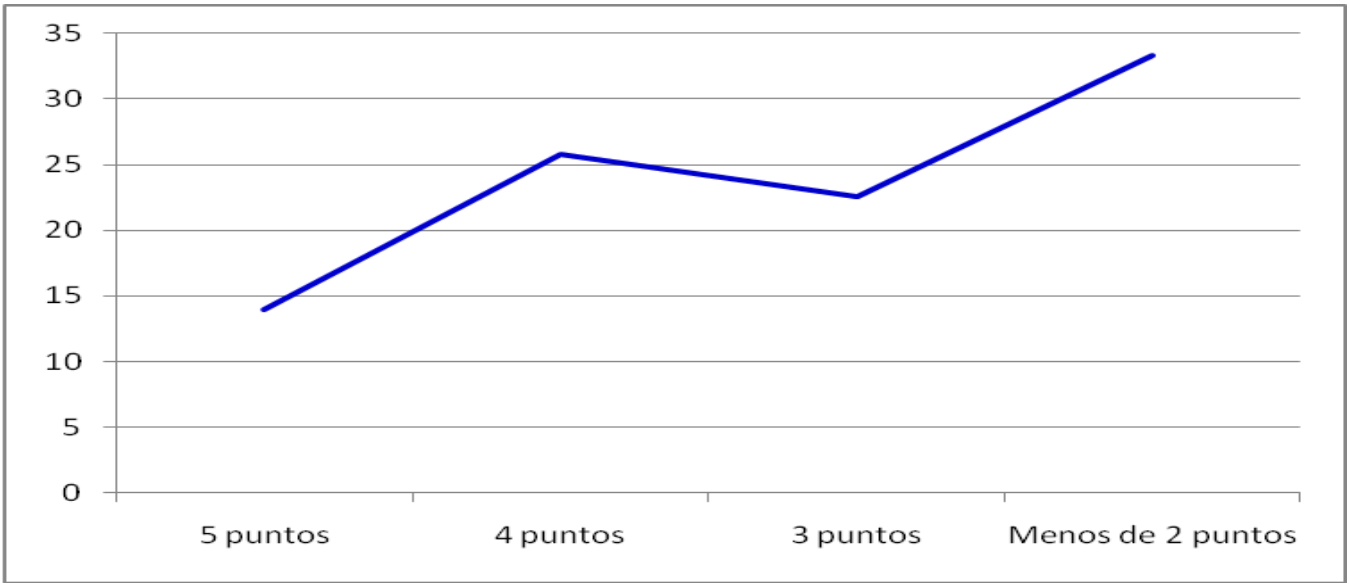


Figura 5. Resultados Diagnóstico.2008-2009.

Al terminar el semestre después de aplicar el plan de estudio H los resultados se comportaron del modo mostrado en la tabla 4 y el gráfico 5:

Tabla 4 Resultados Prueba Final. 2008-2009.

Puntuación	Cantidad de Estudiantes	Porciento del Total
5 puntos	**10**	**11**
4 puntos	**22**	**24**
3 puntos	**30**	**32**
Menos de 2 puntos	**27**	**29**

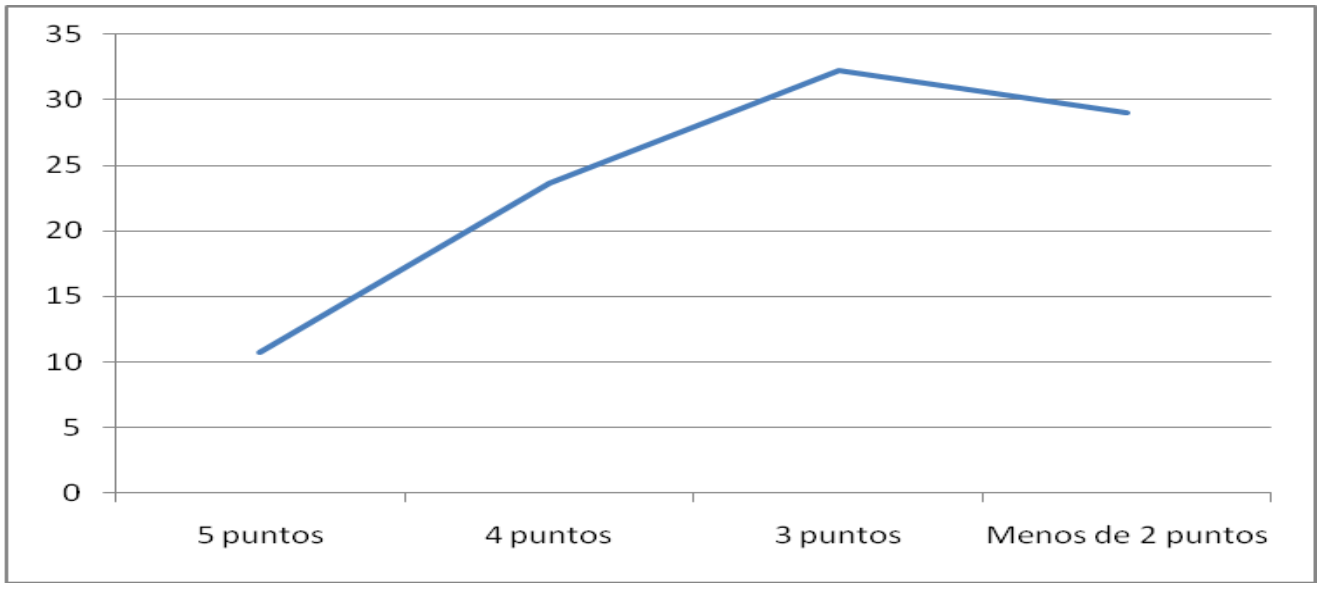


Figura 6. Resultados Prueba Final. 2008-2009.

Si se analizan ambos resultados a partir de la representación en el gráfico 6 se puede concluir que:

Durante el Curso 2008-2009 los estudiantes que obtuvieron resultados excelentes en el diagnóstico inicial disminuyeron al final del curso en un 3%. Los que obtuvieron cuatro puntos experimentaron una disminución del 2 %. Con respecto a la calificación de 3 puntos aumentaron considerablemente en un 10 %. Los desaprobados sólo disminuyeron un 5%.

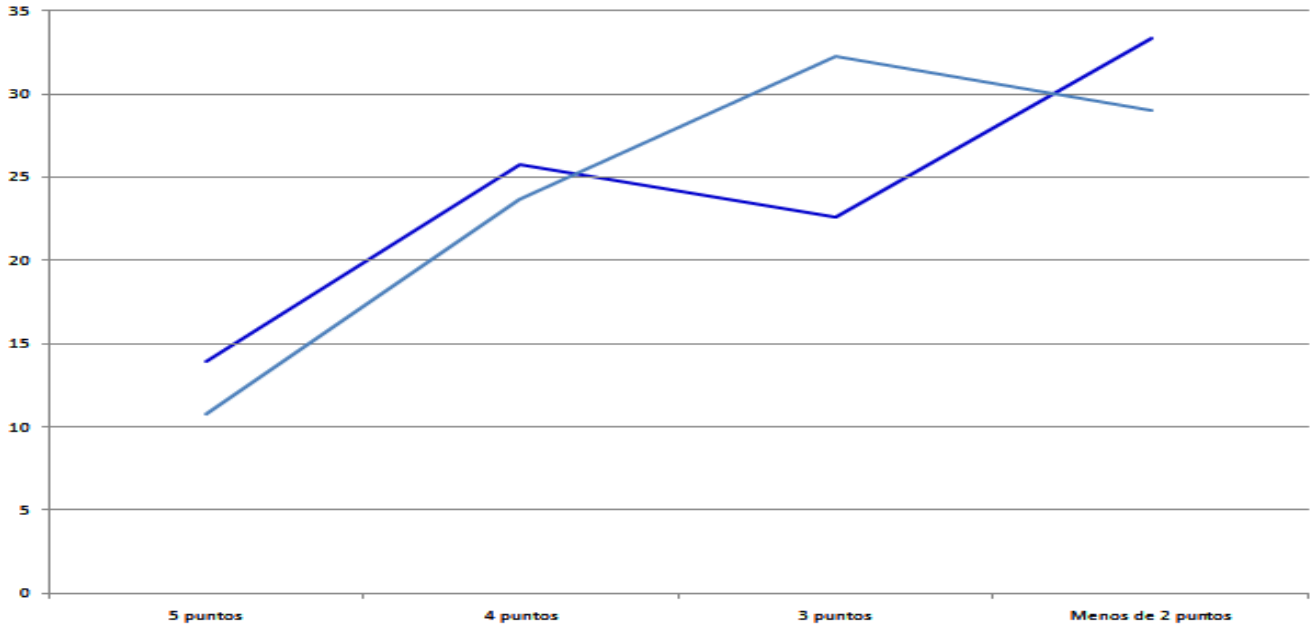


Figura 7. Resultados superpuestos. 2008-2009.

Con el análisis de las estadísticas porcentuales presentadas en este epígrafe se puede intuir que existe algún tipo de problema que está dando al traste con los resultados insatisfactorios que se están obteniendo en la asignatura en cuestión.

1.3.2 Situación Problemática

Al gestarse las primeras ideas del Proyecto Futuro en marzo del 2002 el Comandante en Jefe Fidel Castro Ruz da inicio a un proceso de cambios en la concepción de la enseñanza superior tradicional en Cuba. El 23 de noviembre de 2002 con 2008 estudiantes y más de 300 profesores queda inaugurado el primer curso de la actual UCI.

La UCI es una institución que garantiza la formación profesional de alrededor de 2000 ingenieros en ciencias informáticas desde julio del 2007 con la terminación de cada curso. Con la visión de ser la institución de excelencia que se convierta en el paradigma de las universidades del

país esta tiene la finalidad de informatizar cada sector de la economía desarrollando una industria de software en la isla.

Para logar el desarrollo de los estudiantes, en la ingeniería desde el punto de vista de la formación y la producción, la dirección de tele formación desarrolla e implementa modelos que aplican las Tecnologías de la Información y las Comunicaciones, combinando el protagonismo pedagógico con la aplicación de herramientas tecnológicas avanzadas, contribuyendo a aumentar la calidad del proceso de enseñanza aprendizaje.

Los dos primeros años de la carrera se centran fundamentalmente en la formación básica, asumiéndola con mayor amplitud, incorporando con igual prioridad los aspectos básico-específicos de la carrera, así como otros de carácter más general, indispensables para un profesional en la época actual.

A partir del análisis de los registros de caracterización cualitativos de los cursos 2007-2008 y 2008-2009 y la entrevistas realizadas a más del 50 % de los profesores del departamento de Ciencias Básicas de la facultad 9 se ha podido demostrar que la matemática es una de las disciplinas que muestra mayor grado de problemas en la formación básica de los estudiantes, específicamente las Matemáticas I y III.

Dado que un elevado porciento de estudiantes de nuevo ingreso de los cursos que se analizan reflejan resultados insatisfactorios en el curso de la primera asignatura identificada, según las estadísticas mostradas en el epígrafe 1.3.1, se hizo necesario profundizar en el por qué de esta situación. Teniendo en cuenta lo planteado en las entrevistas y los documentos revisados referentes a este tema se puede concluir que:

- Los estudiantes se caracterizan por ser finalistas y sólo estudian los temas de la asignatura cuando se acercan las pruebas.
- Los estudios independientes que se orientan no se terminan del todo y en algunos casos ni se realizan.
- El acceso a la plataforma virtual de aprendizaje se realiza en gran medida cuando se orienta de manera obligatoria.
- Los estudiantes aventajados no sienten interés por los temas de la asignatura que ya conocen.
- En muchos casos se cree que la asignatura no es necesaria para la carrera.

1.4 Análisis de otras soluciones existentes

En las fuentes consultadas no se hace referencia a SEAI concebidos como un sistema basado en el conocimiento. Se implementan alguno de sus módulos utilizando técnicas de IA.

La literatura estudiada refleja que la tendencia actual está dirigida hacia el desarrollo de modelos para aplicaciones específicas y no modelos generales independiente del dominio de aplicación. Lo que conlleva que el desarrollo de tales sistemas tenga un enfoque multidisciplinario, pues se requiere de conocimiento en el dominio de aplicación, de conocimientos de programación e IA y ellos no siempre coexisten en los especialistas.

La mayoría de los trabajos a los que se hace referencia a continuación están vinculados a un dominio específico, basan su descripción en la arquitectura general del sistema, la información que se almacena, su aplicación, resultados obtenidos y describen la técnica de IA utilizada para implementar el módulo, sin dar detalles del modelo desarrollado, limitando así poder realizar un análisis crítico de los mismos.

Los trabajos de Fernández-Castro, 1989 y VanLehn y Zhendong, 2001 contienen ejemplos de SEAI que utilizan técnicas de IA en el modelo del dominio: marcos y redes bayesianas respectivamente. Los trabajos de Carbonell, 1970, Anderson y Reiser, 1985, André y Finkler, 1993, Prentzas, 2002 y Del Valle, 2004 describen la implementación del módulo pedagógico mediante planes, reglas de producción, redes neuronales artificiales, redes semánticas y redes bayesianas respectivamente.

Otros trabajos se han dedicado al modelo del estudiante entre estos están: (VanLehn y Zhendong, 2001), (Conati, 2002) y (Medina, 2007) redes bayesianas; (Gaudioso, 2002) árboles de decisión; (Cataldi, 2007) redes neuronales artificiales; (Gürer,1995) redes conceptuales (MOP) y (Jiménez, 2008) razonamiento basado en casos y en (González, 2008) se describe la arquitectura de un sistema tutorial para la enseñanza médica que integra sistemas multiagente y el paradigma de aprendizaje basado en casos.

Otros trabajos utilizan la Enseñanza Basada en Casos (EBC) (Schank, 1995). En (Jiménez y Gómez, 2005) se describen aspectos fundamentales de la EBC y son referenciados SEAI tales como: Creanimate (Aleven, 1997), CATO (Edelson, 1996), Proyecto AMBRE (Nogry, 2004), TAO ITS (Stottler, 2001) y JV2M (Gómez-Martín, 2003).

1.4.1 Herramientas de autor para el desarrollo de SEAI

El proceso de producción de Sistemas de Enseñanza-Aprendizaje en cualquiera de sus modalidades es una tarea difícil que requiere mucho esfuerzo por parte de los especialistas de contenido, desarrolladores del currículo, y programadores. De hecho, estuvo fuera del alcance de la mayoría de los maestros, instructores o profesores por mucho tiempo (Deal, 2000).

Este proceso es complejo ya que combina tareas propias de elaboración o diseño de materiales informáticos con otras tareas específicas de la elaboración de materiales educativos. Esta complejidad puede reducirse apoyándose en herramientas apropiadas que automaticen una parte o

todo el proceso de desarrollo de Sistemas de Enseñanza-Aprendizaje (Barchino, 2004), El aumento de la demanda de formación con el uso de las TIC ha propiciado una mayor solicitud y evolución en el desarrollo de Sistemas de Enseñanza-Aprendizaje; potenciando la investigación y el desarrollo, por parte de instituciones, universidades y empresas comerciales, de herramientas cada vez más fáciles de utilizar por los profesores (Brchino, 2004).

Existen varias definiciones de herramientas de autor, y posiciones diversas en cuanto a su denominación (Cataldi, 2009).

En este trabajo se asume la definición dada en (Brchino, 2004) que considera las herramientas de autor como aplicaciones que disminuyen el esfuerzo a realizar por los profesores, maestros, educadores, etc., ofreciéndoles indicios, guías, elementos predefinidos, ayudas y una interfaz amigable para crear materiales educativos y/o Sistema de Enseñanza-Aprendizaje en formato digital.

En general, a pesar de que existen en el mercado un número considerable de herramientas de autor para reducir la complejidad del proceso de elaboración de Sistemas de Enseñanza-Aprendizaje, aún subsisten dificultades en su uso, determinadas por: la complejidad de la interfaz, estar relacionadas con un modelo pedagógico determinado por la plataforma en la cual se entregará software educativo, la realización del proceso por el profesor guiado por estas herramientas y se sigue centrando el problema de la superación del profesor en enseñarlo a usarlas y no a resolver problemas con la ayuda de ellas (Brchino, 2004).

La literatura consultada refleja que la tendencia actual está dirigida hacia el desarrollo de plataformas virtuales de trabajo, tanto en el campo nacional como internacional. Las cuales crean las condiciones básicas para una efectiva gestión del conocimiento, son de inestimable ayuda para lograr extraer y organizar el conocimiento que existe en los más diversos soportes de información, son una oportunidad para evaluar, ampliar y socializar un conocimiento que ya existe y se encuentra disperso, lo que indiscutiblemente conduce a elevar el nivel de conocimiento de la organización y son de gran utilidad en todo el proceso docente-educativo (Cataldi, 2009).

Se pueden citar herramientas para desarrollar Sistemas de Enseñanza-Aprendizaje tales como: Herramienta de autor para generar los cursos del modelo pedagógico tecnológico Microcampus (Laboratorio Multimedia Universidad de Alicante, España), EVA (Entorno Virtual de Aprendizaje), Learning Space, SEPAD (Sistema de Enseñanza Personalizada a Distancia, UCLV), Mundicampus, Moodle, AprenDIST (Plataforma de Teleformación, CUJAE), Atutor, Claroline, WebCT, BlackBoard, Teleduc.

Sin embargo; estas valiosas herramientas de autor no incluyen dentro de sus facilidades la personalización del proceso de enseñanza-aprendizaje; pues no implementan un módulo que contenga las características del estudiante, eludiendo así la existencia de diferentes modelos de estudiantes.

Aunque el avance de herramientas de autor para elaborar SEAI no ha sido el esperado, en la literatura consultada también se encuentran trabajos sobre la existencia de sistemas de autor que permiten al profesorado diseñar cursos adaptativos e interactivos para todas las materias tales como: MAS-PLANG (Macias y Castell, 2001), IRIS-D (Ferrero, 2001). En (Roa, 2005) se describe una mejora a la plataforma Moodle como alternativa para la personalización de los contenidos, PLAGENSTI-IIE (Romero, 2002), (Trella, 2006), SICAD, al que se hace referencia en (González, 2008), ALLEGRO (Jiménez y Ovalle, 2008).

1.4.2 Sistemas Basados en Conocimiento

Los sistemas basados en el conocimiento constituyen técnicas de la IA válidas para enfrentar la construcción de SEAI dado por sus aspectos afines. Estos sistemas utilizan conocimiento sobre un dominio específico. La solución que se obtiene es similar a la obtenida por una persona experimentada en el dominio del problema. Por su parte los SEAI utilizan la información almacenada sobre las características del estudiante para adaptar el proceso de enseñanza-aprendizaje del mismo a la materia a enseñar.

Una característica distintiva de los sistemas basados en el conocimiento es la separación del conocimiento (base de conocimiento) del método de solución del problema (máquina de inferencia). La construcción de la base de conocimiento lleva implícito un arduo proceso de adquisición del conocimiento y es particular para cada sistema, por lo que será necesario construirla para cada aplicación. Sin embargo, la máquina de inferencia puede rehusarse en la construcción de varios sistemas basados en el conocimiento siempre que el tipo de conocimiento y el tipo del razonamiento sea similar.

Diferentes tipos de conocimiento dan lugar a diferentes tipos de sistemas basados en el conocimiento, entre ellos los sistemas basados en reglas (Hand 1997), (Rich 1988), los sistemas basados en probabilidades, (Minka 2001), (Lerner 2002), sistemas expertos conexionistas o redes expertas (Barr y Mani 1994), (Hilera y Martínez 1995) y los sistemas basados en casos (Dutta y Bonissone 1991), (Kolodner 1992).

La máquina de inferencia es el método implementado que utiliza el conocimiento de la base para resolver los problemas del dominio. El tipo de conocimiento determina qué método de solución de problemas es posible utilizar.

No todos los paradigmas para crear sistemas basados en el conocimiento facilitan la concepción de un SEAI, donde lo fundamental para su desarrollo es determinar cómo representar el conocimiento requerido para sus módulos y a partir de dicho conocimiento realizar un diagnóstico del estudiante para que el sistema se adapte a sus características. Sin embargo, similitudes de los SEAI

y los Sistemas Basados en Casos son factores a estudiar para concebir todos los módulos de los SEAI y un diagnóstico adecuado del qué y cómo enseñar dependiendo del estudiante.

1.4.2.1 El Razonamiento Basado en Casos una perspectiva para elaborar SEAI.

El Razonamiento Basado en Casos (RBC), es un enfoque que aborda nuevos problemas tomando como referencia problemas similares resueltos en el pasado. De modo que problemas similares tienen soluciones similares, y la similitud juega un rol esencial. Sus componentes fundamentales son la base de casos, el módulo de recuperación de casos y el módulo de adaptación de las soluciones.

1.4.2.1.1 Base de Casos (BC)

La BC contiene las experiencias, ejemplos o casos a partir de los cuales el sistema hace sus inferencias. Esta base puede ser generada a partir de casos o ejemplos resultantes del trabajo de expertos humanos o por un procedimiento automático o semiautomático que construye los casos desde datos existentes registrados, por ejemplo, en una base de datos.

Puede ser representada a través de una tabla cuyas columnas son etiquetadas por variables o atributos que representan los rasgos predictores y objetivos mientras sus filas representan los casos u otras formas de representación.

1.4.2.1.2 Módulo de Recuperación

En este módulo se recuperan de la Base de Casos los casos más semejantes al problema. No existe una medida de semejanza única, general, para cualquier dominio, de ahí que la eficiencia del sistema radica en la función de semejanza que se defina.

La función de semejanza más sencilla consiste en contar el número de rasgos predictores similares entre ambos, sin embargo se presenta el problema de que la importancia de los rasgos predictores varía de un contexto a otro. La representación más general de función de semejanza pudiera ser:

$$f(O_o, O_t) = \frac{\sum_{i=1}^{n} w_i \cdot \delta_i(x_i(O_0), x_i(O_t))}{\sum_{i=1}^{n} w_i} \qquad (1.1)$$

donde n es el número de rasgos predictores y w_i la importancia asociada al rasgo i .

Ejemplos de funciones de comparación de rasgos:


, para rasgos booleanos (1/0), nominales (ej. B/M/R), numéricos (ej. 5/4/3/2).

$$1 \quad si \quad |x_i - x_t| < \varepsilon$$
$$0 \quad e.o.c$$

, para rasgos numéricos (ej. 5/4/3/2).

$$\delta_i(x_i, x_0, x_t) = ¿\{\}$$

$$\delta_i(x_i, x_0, x_t) = 1 - \frac{|x_i - x_t - x_i - x_o|}{|x_i - x_t + x_i - x_o|}$$, para rasgos numéricos (ej. números reales).

1.4.2.1.3 Módulo de Adaptación

Después de la determinación de los casos más semejantes, las soluciones contenidas en dichos casos pueden usarse directamente como solución al nuevo problema, pero comúnmente necesitan ser modificadas.

El enfoque que utilizan los Sistemas Basado en Casos (SBC) para la adquisición de conocimiento es una de las ventajas que se le acreditan a este tipo de sistemas; pues razonan desde episodios específicos, lo cual evita el problema de descomponer el conocimiento del dominio y generalizarlo en reglas.

Otras de las ventajas de los SBC están fundamentadas; en la flexibilidad para representar el conocimiento a través de los casos, la organización de la BC y de las estrategias de recuperación y adaptación de los casos y que el usuario puede ser capaz de agregar nuevos casos a la BC sin la intervención experta.

Ventajas lo son también, el rehúso de las soluciones previas al resolver un problema, y el almacenar casos que resultó un fracaso, lo que permite advertir sobre problemas potenciales a evitar. Así como también poder fundamentar las soluciones derivadas a partir de casos reales.

Las limitantes de los SBC están en la definición de la función de semejanza y en lo difícil que resulta encontrar una estructura apropiada para describir el contenido de un caso y decidir cómo la memoria de casos debe ser organizada e indexada para un almacenamiento, recuperación y rehúso efectivo.

1.4.2.1.4 Modelo de la Base de Casos

Un razonador basado en casos depende de la estructura y el contenido de la base de conocimiento. Hasta el presente, para resolver el problema de organizar una BC, un enfoque ha sido

almacenar los casos de forma secuencial y analizarlos todos para resolver el nuevo problema. Este tipo de organización hace lento el proceso de recuperación.

Un método alternativo consiste en particionar los casos en grupos y organizarlos jerárquicamente. Esta jerarquía permite una búsqueda más eficiente ya que se sigue por un determinado camino en dependencia de los valores de los rasgos predictores del nuevo problema.

1.4.2.1.5 Selección de rasgos en los Sistemas Basados en Casos.

La selección de rasgos es cuestión central tanto en la definición del modelo de la base de casos como en el modelo de recuperación de casos. Potencialmente, en el conjunto de casos podrían estar todas las propiedades que describen los objetos; pero existen rasgos inútiles que carecen de importancia de acuerdo al dominio de aplicación.

El conjunto de rasgos determina qué información será almacenada en memoria para cada uno de sus elementos, la cual debe permitir la posterior recuperación de los casos semejantes dada la descripción de un nuevo problema. Otro aspecto es que dentro del conjunto de rasgos que se seleccionan no todos tienen la misma importancia y esta diferencia debe tenerse en cuenta para comparar objetos.

En los SBC dentro de los criterios más comúnmente utilizados en el cálculo de la importancia de cada rasgo están: criterio de los especialistas del dominio de aplicación, dispersión de los valores del rasgo, frecuencia del valor dado al rasgo, carácter diferenciante del rasgo, fuerza predictiva del rasgo, entre otros.

De forma general el problema de selección de rasgos consiste en encontrar el subconjunto de rasgos que mejor describe los objetos del dominio; usualmente este subconjunto se encuentra maximizando o minimizando una función objetivo.

En el enfoque lógico combinatorio una alternativa de solución al problema de la selección de rasgos es a partir de la utilización del conjunto de testores típicos. En esencia, un testor es un conjunto de características (rasgos) que diferencia a elementos (objetos) de clases distintas.

Los testores típicos constituyen variantes minimales de subconjuntos de rasgos, existiendo algoritmos que permiten calcular la importancia del rasgo a partir de estos.

Para el cálculo de los testores típicos se han desarrollado varios algoritmos que por su estrategia de cómputo pueden clasificarse en algoritmos de escala exterior y de escala interior.

El enfoque lógico combinatorio es una alternativa a tener en cuenta en el modelo que se propone en este trabajo para la selección de los rasgos predictores relevantes y a partir del conjunto de los testores típicos calcular la importancia de los rasgos predictores. La selección de este enfoque está estrechamente relacionada con las características propias del sistema a desarrollar, dígase

número de casos, número de rasgos predictores, dominio de definición de los rasgos predictores entre otras.

1.5 Conclusiones Parciales

En el presente capítulo:

- Se conceptualizó la definición de un SEAI enfatizándose en los aspectos comunes con los sistemas basados en el conocimiento.
- Se analizaron los diferentes tipos de sistemas basados en el conocimiento y los problemas actuales de estos sistemas relacionados con la adquisición y representación de la información.

A partir del análisis realizado, se plantea:

- La factibilidad de desarrollar un modelo que permita implementar los tres módulos fundamentales de un SEAI en la BC y realizar el modelado del estudiante utilizando el paradigma del RBC.
- La utilidad de desarrollar un SEAI por usuarios no necesariamente expertos en computación; sin suponer una carga excesiva de trabajo adicional a la tarea ya de por sí considerable de desarrollar un SEAI.

CAPÍTULO 2

Herramientas y Tecnologías a utilizar.

2.1 Introducción

En el presente capítulo se describen los principales elementos a tener en cuenta para la implementación del Sistema de Enseñanza Aprendizaje de Apoyo al curso introductorio de Matemática. Esta información está dirigida a la fase de construcción donde se plantean los lenguajes a utilizar, las herramientas y la metodología que controla la organización y la calidad de la implementación.

2.2 Metodologías de Desarrollo de Software

En la actualidad las metodologías de desarrollo de software pueden ser clasificadas en dos categorías: ágiles y tradicionales (Jacobson, 2000).

2.2.1 Un enfoque Tradicional.

El Proceso Unificado (RUP) se proclama dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental. Como casi todas las metodologías es el resultado de más de treinta años de experiencia.

RUP junto con el Lenguaje Modelado Unificado (UML), constituye una metodología estándar utilizada para el diseño, implementación y documentación de los sistemas orientados a objetos. Más allá de establecer un conjunto de pasos a seguir RUP es un conjunto de metodologías adaptables a las necesidades de cada organización.

RUP es un producto de IBM que está basado en 5 principios fundamentales:

- Adaptar el proceso: El proceso debe adaptarse a las características propias del proyecto u organización.
- Balancear Prioridades: Los requerimientos de los diversos inversores pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos.
- Demostrar valor iterativamente: Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas.

- Elevar el nivel de abstracción: Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes o esquemas (frameworks), por nombrar algunos.
- Enfocarse en la calidad: El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción.

RUP es más apropiada para proyectos grandes, dado que requiere un equipo de trabajo capaz de administrar un proceso complejo en varias etapas. En proyectos pequeños, es posible que no sea posible cubrir los costos de dedicación del equipo de profesionales necesarios. (Jacobson, 2000)

2.2.2 Un enfoque Ágil

En el 2001, motivados por la observación de que en muchas compañías los equipos de software estaban atascados cada vez más en el lodo de los procesos, un grupo de expertos de la industria se reunieron para establecer los valores y principios que permitan a los equipos de software trabajar rápidamente y responder al cambio. Ellos se denominaron a sí mismos la Alianza Ágil. Durante varios meses este grupo trabajó para crear una declaración de valores. El resultado es El Manifiesto de la Alianza Ágil. (Jacobson, 2000)

Manifiesto para el Desarrollo de Software Ágil

Se está descubriendo mejores maneras de desarrollar Software haciéndolo y ayudando a otros a hacerlo. A través de este trabajo se valora:

- Los individuos y las interacciones sobre los procesos y las herramientas.
- Software operativo sobre documentos detallados.
- Colaboración del cliente sobre la negociación de contratos.
- Responder a los cambios sobre seguir un plan.” (Martínez, 2005)

El enfoque Ágil considera que el problema básico del desarrollo de software es el riesgo, consecuente con la premisa de incertidumbre inevitable. Para enfrentar el riesgo, el enfoque Ágil propone ciclos cortos de versiones y dentro de ellas, iteraciones de una a cuatro semanas; implementar las características de más alta prioridad; integrar al cliente al equipo de desarrollo; crear y mantener conjuntos detallados de pruebas, el enfoque Ágil, renuncia a la anticipación a escala de iteración y de sistema. Adopta una actitud reactiva, en el sentido de Roger S Pressman. También estima que el diseño no es dibujar un puñado de esquemas y entonces implementar el sistema conforme a esos esquemas. Se reconoce que los esquemas pueden aportar beneficios, pero que no ofrecen una retroalimentación concreta, es decir de funcionamiento. Con esta última idea, el enfoque Ágil marca profundamente su vocación exploratoria y rompe el dogma que exige un buen diseño antes de la implementación. Una vez más se hace patente la premisa de incertidumbre inevitable que no se puede eliminar por mucho que se piense.

El análisis de las propiedades del enfoque Ágil lo aleja de las metodologías tradicionales, más allá del Proceso Unificado, en el eje de incertidumbre. El enfoque Ágil aplica una estrategia cíclica, de mayor frecuencia, con una componente exploratoria importante. Se podría decir que el enfoque Ágil se dirige a problemas del tipo “sabré lo que quiero cuando lo vea”, referidos por Barry Boehm para justificar la Espiral (Martínez, 2005).

La capacidad de incertidumbre del modelo de objetos, su moderada inercia, se acomoda relativamente bien a las cualidades de movilidad del enfoque Ágil, aunque convendría un modelo menos inerte aún para aumentar la eficiencia del ensayo y error. El modelo de software tradicional (funciones y datos) es demasiado inerte, aún cuando renuncie a los esquemas (dibujos), a causa de su riqueza de relaciones unívocas.

2.2.3 Human Computer Interface (HCI)

La HCI es una disciplina que a pesar de registrar sus inicios en la explosión tecnológica de los años setenta, es poco conocida en el área informática. Aunque existe cierto interés por parte de los diseñadores, no se desarrolla de forma sistematizada en países como Cuba. Esto viene dado por la cobertura ínfima y casi risible del tema en la bibliografía existente y en los productos que la aplican al pie de la letra. Esta especialidad se vale de áreas del conocimiento como la psicología cognitiva, la informática, la lingüística y la ingeniería del diseño.

Denominada en sus orígenes como Man Machine Interaction (MMI), la HCI se ocupa del análisis y diseño de interfaces de usuario que estudian y buscan poner en práctica procesos orientados a la construcción de interfaces siguiendo el criterio de usabilidad, es decir con alto grado de facilidad en el uso del sistema interactivo de acuerdo al estándar ISO 92401 de requisitos ergonómicos para el trabajo de oficina con terminales visuales y normas asociadas. Se basan en aplicación de las leyes gestálticas que están relacionadas con los criterios de Smith y Mosier, 1992) y las normas ISO 9241 (1998) y 11064 (2000) para el diseño de interfaces y ergonomía (Zulma Cataldi, 2009).

La ergonomía en Europa y Human Factors en EUA, nació durante la Segunda Guerra Mundial con el objetivo de diseñar armamento militar cómodo de usar. En los años 60 comenzaron a llevarse estos estudios al ámbito de la informática para el diseño de interfaces en pantalla. Se basa en realzar la calidad del uso de los objetos, en maximizar la comodidad y la eficiencia para hacer más fáciles las tareas y aumentar el confort y la satisfacción. Para mejorar la HCI se ha centrado especialmente en el hardware (monitores, teclados, ratones, etc.), si bien también trata aspectos de software que afectan a la psicología, como es la legibilidad en pantalla.

La Organización Internacional de Normalización (ISO) ha asignado a varios comités la elaboración de normas sobre ergonomía pero el TC159/SC5 es el encargado de la ergonomía en la interacción hombre-máquina.

El resultado más notable por el momento ha sido la norma ISO 9241, dedicada a normativas sobre diseño (Marcos, 2001).

Para el desarrollo de un STI, después de analizar con profundidad todos los puntos expuestos anteriormente sobre HCI, se convierten en un potente instrumento casi de obligatorio cumplimiento los principios básicos de usabilidad conceptualizados por la ISO:

- Facilidad de Aprendizaje: facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema o producto.
- Flexibilidad: relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.
- Robustez: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos. Está relacionada con la capacidad de observación del usuario, de recuperación de información y de ajuste de la tarea al usuario.

¿Cómo asegurar la usabilidad del producto?

La usabilidad no es un detalle que pueda ser aplicado en el último minuto sino que debe tomarse en cuenta desde el inicio de cualquier proyecto. Durante el proceso de desarrollo de un software cualquier decisión de análisis, diseño o implementación pueden afectar profundamente este aspecto

Siguiendo las normas de la Parte 11 del estándar que se sometió a análisis en el presente epígrafe, se asegura la usabilidad del software SEAI para el curso introductorio de matemática incorporando a los usuarios en todas las iteraciones del proceso; estableciendo medidas cuantitativas desde el principio para hacer test de usabilidad, trabajando en la medida de las posibilidades con un enfoque multidisciplinario y aplicando una metodología de desarrollo iterativo.

El análisis de los principios planteados tanto por las metodologías ágiles como por las tradicionales aportan una visión clara de que con un proyecto de tesis como el que se quiere desarrollar para obtener el producto: SEAI para el curso introductorio de matemática, no es necesario seguir los pasos de RUP como es típico en la UCI. En primer lugar, sólo se cuenta con dos desarrolladores de modo que es un equipo pequeño. El cliente según los propios principios de los SBC adopta el papel de experto en este caso convirtiéndose en un trabajador más del equipo. Las ideas innovadoras del producto que se planea obtener exigen un avance en todas las áreas que

contribuyan a su satisfactoria terminación por lo que la metodología no queda exenta. En resumen, se decidió la utilización de RUP ágil (AUP 1.1) para el proyecto en cuestión así como los principios de Human Computer Interaction (HCI) (Normas ISO 92401) para el diseño de la interfaz.

2.3 Herramientas CASE

Se puede definir a las Herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software. Como es sabido, los estados en el ciclo de vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación Prueba e Instalación básicamente.

Ejemplos de herramientas CASE se puede mencionar Rational Rose y Visual Paradigm, este último es utilizado para el desarrollo del presente trabajo por ser multiplataforma.

2.3.1 Visual Paradigm 6.4 para UML

Es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste además de ser multiplataforma. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML (Medina Pasaje, 2006).

2.4 Lenguajes de Programación.

Un Lenguaje de Programación es, técnicamente hablando, un conjunto de instrucciones que son entendibles y ejecutables por un computador. No se puede esperar, por lo menos no por ahora, que el computador ejecute lo que se concibe como algoritmo (aunque sería lo óptimo) y por eso se debe incorporar al desarrollo técnico un paso o un conjunto de pasos más y son lo que involucran los Lenguajes de Programación.(Medina Pasaje, 2006)

Esto significa que si se ha desarrollado un algoritmo computacional que es muy útil pues de nada va a servir si se conocen las reglas sintácticas de un Lenguaje de Programación, si no se escribe el algoritmo con esas reglas, si no se cuenta con el compilador del Lenguaje en el cual se va a trabajar y si no se sabe interpretar los errores. Por lo tanto el contacto técnico que se va a tener con el computador va mucho más allá de desarrollar solo el algoritmo ya que es obligación hacer realidad el algoritmo mediante el lenguaje de programación.

2.4.1 PHP 5 (Acrónimo de Hypertext Preprocessor)

Como lenguaje del lado del servidor, esto significa que PHP funciona en un servidor remoto que procesa la página Web antes de que sea abierta por el navegador del usuario. Este lenguaje especialmente está creado para el desarrollo de páginas Web dinámicas, es por esto que está dotado de un gran número de funciones que simplifican enormemente tareas habituales como descargar documentos, enviar correos, trabajar con cookies y sesiones, etc. Otras características fundamentales del lenguaje es su compatibilidad con los sistemas operativos basados en UNIX (Linux, Solares, FreeBSD), como con Windows, el sistema operativo de Microsoft y su sencilla integración con múltiples bases de datos como son MySQL, PostgreSQL, Oracle, dbm, filePro, interbasem o cualquier otra base de datos compatible con ODBC (Open Database Connectivity Standard) (Vázquez, 2003).

Entre sus características fundamentales están:

- Gratuito: Al tratarse de software libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.
- Gran popularidad: Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código.
- Enorme eficiencia: Con escaso mantenimiento y un servidor gratuito, puede soportar sin problema millones de visitas diarias.
- Sencilla integración con múltiples bases de datos: Puede conectarse a PostgreSQL, Oracle, dbm, filePro, interbasem o cualquier otra base de datos compatible con ODBC.
- Versatilidad: PHP puede usarse con la mayoría de sistemas operativos, ya sea basados en UNIX (Linux, Solares, FreeBSD), como con Windows, el sistema operativo de Microsoft.
- Gran número de funciones predefinidas: A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas Web dinámicas. Por ello, está dotado de un gran número de funciones que simplifican enormemente tareas habituales como descargar documentos, enviar correos, trabajar con cookies y sesiones (Vázquez, 2003)

El principal objetivo de PHP5 ha sido mejorar los mecanismos de Programación Orientada a Objeto para solucionar las carencias de las anteriores versiones de PHP.

2.4.2 HTML como lenguaje del lado del cliente

HTML es un lenguaje de etiquetas (también llamado lenguaje de marcado) y las páginas web habituales están formadas por cientos o miles de pares de etiquetas. De hecho, las letras "ML" de la sigla HTML significan "markup language", que es como se denominan en inglés a los lenguajes de marcado. Además de HTML, existen muchos otros lenguajes de etiquetas como XML, SGML, DocBook y MathML (Rápida, 2007).

La principal ventaja de los lenguajes de etiquetas es que son muy sencillos de leer y escribir por parte de las personas y de los sistemas electrónicos. La principal desventaja es que pueden aumentar mucho el tamaño del documento, por lo que en general se utilizan etiquetas con nombres muy cortos.

Las páginas HTML se dividen en dos partes: la cabecera y el cuerpo. La cabecera incluye información sobre la propia página, como por ejemplo su título y su idioma. El cuerpo de la página incluye todos sus contenidos, como párrafos de texto e imágenes (Rápida, 2007)

Definiéndolo de forma sencilla, HTML es lo que se utiliza para crear todas las páginas web de Internet. Más concretamente, HTML es el lenguaje con el que se "escriben" la mayoría de páginas web (Rápida, 2007).

2.5 Sistemas Gestores de Bases de Datos

El sistema manejador de bases de datos es la porción más importante del software de un sistema de base de datos. Database Manager System (DBMS) es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

Las funciones principales de un DBMS son:

- Crear y organizar la base de datos.
- Establecer y mantener las trayectorias de acceso a la base de datos de tal forma que los datos puedan ser accedidos rápidamente.
- Manejar los datos de acuerdo a las peticiones de los usuarios.
- Registrar el uso de las bases de datos.
- Interacción con el manejador de archivos.

El manejador de base de datos es el responsable del verdadero almacenamiento de los datos, a través de las sentencias de un lenguaje de manipulación de datos (Data Manipulation Language DML) al comando del sistema de archivos (García, 1999).

2.5.1 Postgre SQL 8.2

PostgreSQL es un sistema gestor de base de datos objeto-relacional (RDBMS). Soporta subconsultas, procedimientos almacenados, tipos de datos complejos como estructuras geométricas espaciales, datos del tipo "direcciones IP" y matrices en una sola celda de una tabla e integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos. Sin embargo, carece de potencia en replicación, distribución, OLAP, etc., temas que son de amplio dominio de las RDBMS comerciales. No tiene toda la documentación completamente traducida. Tampoco tiene servicios avanzados de desfragmentación ni reordenado y en cierto modo se parece

a Visual FoxPro en el manejo de las tablas y los índices, se percibe como una base de datos de escritorio a la que le han colocado una interfaz cliente-servidor (Worsley, 2002).

PostgreSQL cumple completamente con las características ACID (acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español) para realizar transacciones seguras, es multiplataforma, está disponible para 34 plataformas en su última versión estable. Con la integridad referencial y posee interfaces nativas para lenguajes como ODBC, JDBC, C, C++, PHP, PERL, TCL, ECPG; PYTHON y RUBY, además de traer soporte para la herencia y la seguridad de la capa de dispositivo de transportación de datos (SSL, Secure Sockets Layer) (Worsley, 2002).

2.6 Servidor Web Apache 2.2

El servidor web Apache se ha convertido en el servidor web más utilizado en el mundo debido a sus altas prestaciones y desempeño, además de ser gratuito, lo cual contribuye a su rápida expansión y posicionamiento. La configuración de este servidor web para aquellas personas que posean un conocimiento medio del sistema operativo Linux no debe ser un problema, pero resulta en ocasiones complicado e intimidante enfrentarse a los archivos de configuración del servidor sin una guía o con la base de la información fragmentada y de lenguaje oscuro que se puede obtener en la web.

Tiene como ventajas fundamentales:

- Su licencia es de código abierto del tipo BSD que permite el uso comercial y no comercial de Apache.
- Una talentosa comunidad de desarrolladores siguiendo un proceso abierto de desarrollo.
- Arquitectura modular. Los usuarios de Apache pueden adicionar fácilmente funcionalidad a sus ambientes específicos.
- Portabilidad. Apache trabaja sobre todas las versiones recientes de UNIX y Linux, Windows, BeOs, mainframes.
- Es robusto y seguro (Márquez Díaz, 2002).

Por ser gratuito, Apache es uno de los servidores de Web más utilizados y que presenta garantías suficientes para el montaje de sitios Web confiables tanto a nivel de organizaciones independientes y para el ofrecimiento de servicios de hosting a otras organizaciones o en la misma organización a través de los servidores virtuales (Márquez Díaz, 2002)

2.7 Framework Symfony 1.2.9

Un framework simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además proporciona estructura al

código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, un framework facilita la programación de aplicaciones, ya que encapsula operaciones complejas en instrucciones sencillas.

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

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows (Potencier, 2008)

Symfony puede ser completamente personalizado para cumplir con los requisitos de las empresas que disponen de sus propias políticas y reglas para la gestión de proyectos y la programación de aplicaciones. Por defecto incorpora varios entornos de desarrollo diferentes e incluye varias herramientas que permiten automatizar las tareas más comunes de la ingeniería del software:

- Las herramientas que generan automáticamente código han sido diseñadas para hacer prototipos de aplicaciones y para crear fácilmente la parte de gestión de las aplicaciones.
- El framework de desarrollo de pruebas unitarias y funcionales proporciona las herramientas ideales para el desarrollo basado en pruebas ("test-driven development").
- La barra de depuración web simplifica la depuración de las aplicaciones, ya que muestra toda la información que los programadores necesitan sobre la página en la que están trabajando.
- La interfaz de línea de comandos automatiza la instalación de las aplicaciones entre servidores.
- Es posible realizar cambios "en caliente" de la configuración (sin necesidad de reiniciar el servidor).
- El completo sistema de log permite a los administradores acceder hasta el último detalle de las actividades que realiza la aplicación (Potencier, 2008).

2.8 Conclusiones

En este capítulo se argumenta el por qué del uso de las herramientas y metodologías a utilizar en el desarrollo del trabajo. Para la elaboración de la aplicación se escogió como metodología a

seguir Agile UP y los principios de HCI para el desarrollo de la interfaz. La herramienta de modelado es Visual Paradigm, el lenguaje de programación PHP y del lado del cliente HTML, el Sistema Gestor de Bases de Datos: Postgre SQL, el servidor Web: Apache y como marco de trabajo Symfony.

CAPÍTULO 3

Presentación de la solución propuesta.

3.1 Introducción

En este capítulo se describe el negocio utilizando Unified Model Language (UML) 2.0 como lenguaje de modelado y Agile Unified Process (AUP) como Metodología, empleando uno de los artefactos que brinda: el Modelo del Dominio. El propósito del modelo del dominio en este caso es comprender y describir las clases más importantes dentro del contexto del sistema a desarrollar. Sólo se definen los principales conceptos relacionados con el entorno del problema y se presenta un diagrama de los mismos así como una breve descripción de las clases identificadas.

Se muestra una síntesis de la especificación de requisitos del sistema así como del estado del modelo de casos de uso del sistema en su primera versión. Se resumen los principales actores del sistema así como los casos de uso identificados hasta el momento. Esta parte recoge todo lo relacionado con el sistema, desde la definición de los actores del sistema hasta las especificaciones de los casos de uso determinados y los diagramas que sirven de apoyo al trabajo de los desarrolladores así como la arquitectura en la que estará sustentado.

3.2 Modelo de Dominio

3.2.1 Diagrama de clases del Modelo de Dominio

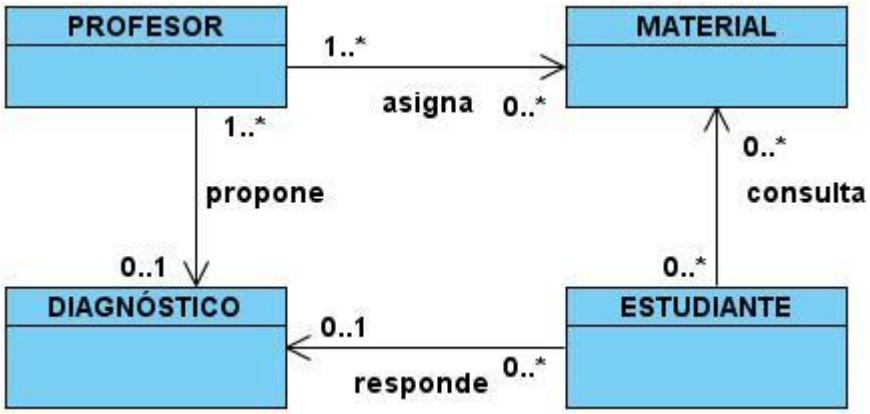


Figura 8. Diagrama de clases del Modelo de Dominio.

3.2.2 Definición de las clases del modelo del dominio

3.2.2.1 Profesor

El profesor es la persona que se encarga de aplicar los diagnósticos y pruebas a los estudiantes a partir del contenido que él planifica e imparte durante el curso de Matemática.

3.2.2.2 Estudiante

El estudiante es quien recibe el contenido durante el curso y se evalúa a partir de las pruebas planteadas por el profesor.

3.2.2.3 Diagnóstico

El diagnóstico es la primera prueba que se realiza al estudiante a modo de evaluación, donde este responde las preguntas a partir de los conocimientos básicos que posee de la materia en cuestión. El test es propuesto por el profesor.

3.2.2.4 Material

El material es toda la documentación que se le asigna al estudiante por parte del profesor para que sea consultado durante su estudio individual.

3.3 Especificación de Requisitos.

Según el estándar 1233 e la IEEE: Guía para el desarrollo de Especificaciones de Requerimientos de Sistemas, un requerimiento se define como:

> Condición o Capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
>
> Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente. (IEEE, 1998)

Todas las ideas que los clientes, usuarios y miembros del equipo de proyecto tengan acerca de lo que debe hacer el sistema, deben ser analizadas como candidatas a requisitos.

3.3.1 Requisitos Funcionales

RF1: Autenticar Usuario.

RF2: Gestionar Usuario.

2.1 Insertar Usuario.

2.2 Modificar Usuario.

2.3 Eliminar Usuario.

RF3: Gestionar Rol.

3.1 Insertar Rol.

3.2 Modificar Rol.

3.3 Eliminar Rol.

RF4: Consultar Material.

RF5: Realizar Test.

RF6: Gestionar Caso.

6.1 Insertar Caso.

6.2 Modificar Caso.

6.3 Eliminar Caso.

RF7: Gestionar Material.

7.1 Insertar Material.

7.2 Modificar Material.

7.3 Eliminar Material.

RF8: Gestionar Cuestionario.

8.1 Insertar Cuestionario.

8.2 Modificar Cuestionario.

8.3 Eliminar Cuestionario.

RF9: Gestionar Pregunta.

9.1 Insertar Pregunta.

9.2 Modificar Pregunta.

9.3 Eliminar Pregunta.

RF10: Obtener Reporte.

3.3.2 Requisitos No Funcionales

3.3.2.1 Usabilidad

El sistema estará dirigido a estudiantes con cualquier nivel de conocimiento en matemática básica y a profesores interesados en conocer el comportamiento de los estudiantes en el Software Educativo SEAI-Matemática. El mismo será utilizado por:

- Usuarios registrados, estos pueden ser administradores o profesores, a los cuales se les asignan privilegios.
- Si el usuario es un profesor puede tener acceso a los datos de los diferentes casos almacenados y al manejo de los materiales disponibles.
- Estudiantes preferiblemente de primer año que cursan Matemática I.

La herramienta debe ser multiplataforma.

El software debe implementarse de manera que se asegure que luego de su instalación se ejecute con calidad en cualquier plataforma.

Brindar Ayuda en Línea.

El sistema está diseñado para ser utilizado por personas con conocimientos medios en el manejo de la computadora y el ambiente Web en sentido general, debido a ello se cuenta con una ayuda del sistema a fin de documentar al usuario en la utilización de la herramienta. La ejecución de las acciones debes ser posible por el uso del teclado u otro dispositivo como el mouse.

Brindar reportes de errores de manera dinámica.

Los mensajes de error deben ser reportados por la propia aplicación en la medida de las posibilidades y no por el Sistema Operativo. Los mensajes del sistema deben estar en el idioma apropiado, en este caso español.

3.3.2.2 Fiabilidad

La fiabilidad es sólo la probabilidad de que el sistema funcione bien en un período de tiempo determinado y bajo ciertas condiciones.

El sistema debe ser capaz de mantener la calidad de los datos de manera que garantice su integridad durante su procesamiento.

3.3.2.3 Soporte

Los servicios de instalación y mantenimiento del sistema será responsabilidad del administrador del sistema en la entidad que sea utilizado.

El sistema debe contar con la capacidad de ser escalable

La escalabilidad del sistema debe asegurarse desde la arquitectura sobre la cual se implemente hasta las herramientas usadas para ello.

3.3.2.4 Restricciones de diseño

A continuación se especifica la construcción del un sistema a partir de algunas restricciones que han sido ordenadas y deben ser cumplidas estrictamente.

Usar estándares de codificación PHP.

Utilizar lenguaje PHP 5 para la programación del producto.

La arquitectura escogida debe dar soporte a los módulos que incluye la solución planteada para el desarrollo del Sistema.

3.3.2.5 Restricciones de seguridad.

La seguridad de un sistema solo se tiene en cuenta desde el punto de vista del ingeniero de software. Por lo que no se contempla la seguridad física del lugar donde se podría usar la aplicación ya que no se incluye en el alcance de este trabajo en general.

Protección contra acceso no autorizado.

La información es protegida contra accesos no autorizados utilizando mecanismos de autenticación y autorización que puedan garantizar el cumplimiento de esto: cuenta, contraseña y nivel de acceso, de manera que cada uno pueda tener disponible solamente las opciones relacionadas con su actividad y tenga datos de acceso propios, garantizando así la confidencialidad. No obstante los usuarios accederán de manera rápida y operativa al sistema sin que los requerimientos de seguridad se conviertan en un retardo para ellos.

Encriptar Datos

Se usan mecanismos de encriptación de los datos que por cuestiones de seguridad no deben viajar al servidor en texto claro, como es el caso de las contraseñas. Se usa como algoritmo de encriptación MD5.

Realizar validación de datos.

Se realizan validaciones de la información tanto en el cliente como en el servidor.

Implementar niveles de acceso

Se crean usuarios con diferentes niveles de acceso al sistema.

3.3.2.6 Restricciones de rendimiento

El sistema deberá ser rápido ante las solicitudes de los usuarios y en el procesamiento de la información.

Los tiempos de respuesta del sistema son de corta duración.

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

El sistema tendrá ayuda en la que se realizarán ciertas aclaraciones sobre algunas opciones del sistema que garantizarán el buen desempeño de los usuarios a la hora de interactuar con el mismo.

3.3.2.7 Interfaz

El diseño del sistema debe estar enfocado en una interfaz sencilla y funcional, de fácil operación, basada en ventanas y de rápida adaptación para los usuarios.

- Interfaces de usuario
- Test para estudiantes
- Interfaz de administración
- Interfaz para uso de profesores

3.3.2.8 Interfaces Hardware

Las terminales clientes solo requerirán de una computadora conectada a la red, para poder ejecutar los navegadores de Web al menos deben cumplir los requisitos mínimos (que requiera el navegador en cuestión).

3.3.2.9 Interfaces Software

La aplicación debe poderse ejecutar en entornos Windows, Linux, etc. (Multiplataforma), para su ejecución del lado del servidor necesita Apache como servidor Web, del lado del cliente cualquiera de los exploradores Web existentes en el mercado.

3.3.2.10 Requisitos de Licencia

El Sistema debe cumplir con los lineamientos necesarios para la producción de software libre y la comunidad de desarrollo y soporte.

3.3.2.11 Requisitos legales, de Derecho de autor y otros

La aplicación debe cumplir con lineamientos, políticos y/o regulaciones de la UCI como entidad que potencialmente utilice el sistema.

3.3.2.12 Estándares Aplicables.

En este trabajo se siguen los principios establecidos en el estándar ISO 9241-11 referentes a la usabilidad en sistemas de software.

3.4 Descripción de la solución propuesta.

Dándole cumplimiento a los requerimientos antes expuestos se ha analizado la creación de un sistema que posibilite, a los profesores vinculados al proceso de enseñanza de Matemática I, medir el desempeño de los estudiantes en la misma. Para ello, los usuarios del sistema han de poseer un conjunto de características, es decir deben ser registrados en la aplicación, para luego darle los privilegios que según su categoría le correspondan, una vez realizada esta operación serán entonces clientes del sistema de enseñanza aprendizaje inteligente propuesto para la asignatura.

Como parte de la gestión de la información generada en este proceso de atención a los estudiantes, los profesores podrán conocer una serie de reportes que serán generados por la aplicación según las necesidades que estos presenten, además de poder consultar e interactuar con la información que se arroje del proceso de atención personalizada.

Tabla 5 Descripción de los actores del sistema.

Actor	Descripción
Profesor	Interesado en conocer el desempeño del estudiante en el sistema.
Estudiantes	Interactúa con el sistema usándolo como herramienta de aprendizaje adaptable a su nivel de conocimiento en la asignatura Matemática.

Administrador	Encargado de la asignación de permisos y el estado del sistema en general.

3.4.1 Casos de Uso del Sistema

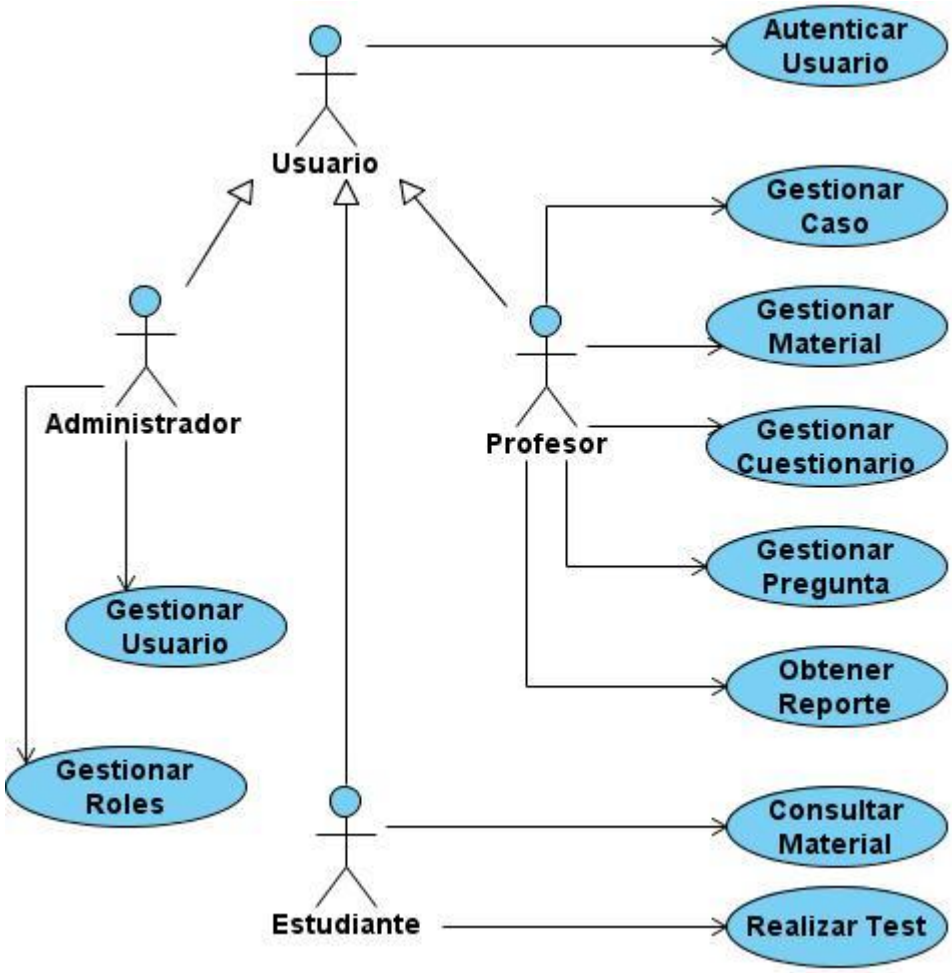


Figura 9. Diagrama de Casos de Uso del Sistema.

3.4.2 Especificación de los Casos de Uso del Sistema

Tabla 6 Descripción del CUS Autenticar Usuario.

Caso de Uso 1:	Autenticar Usuario.
Actores:	Usuario.
Resumen:	El caso de uso se inicia cuando alguno de los actores necesita usar la aplicación para realizar alguna acción con la misma. Accede usando su usuario y contraseña del dominio UCI.
Precondiciones:	1. Ser Usuario Autorizado. 2. Contar con la Aplicación en modo “disponible”.
Referencias	RF 1.

<table>
<tr><td>Prioridad</td><td colspan="2">Crítico.</td></tr>
<tr><td colspan="3">Flujo Normal de Eventos</td></tr>
<tr><td colspan="3">Sección “1”</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1-Abrir Aplicación.</td><td colspan="2">2- Solicitar Autenticación.</td></tr>
<tr><td>3- Introducir usuario y contraseña.</td><td colspan="2">4- Si los datos son correctos se brinda acceso a las diferentes vistas de la aplicación en dependencia de los derechos del usuario definidos por su rol.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
<tr><td colspan="3"> 
</td></tr>
<tr><td colspan="3">Flujos Alternos</td></tr>
<tr><td>Acción del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td>1. Introducir datos incorrectos.</td><td colspan="2">2. Se muestra un mensaje de error en los datos: “Usuario o contraseña incorrectos”, y redirecciona al usuario a la página de autenticación.</td></tr>
<tr><td>3. Aceptar sin haber llenado los campos previamente.</td><td colspan="2">4. El sistema muestra el mensaje “Usuario o contraseña incorrectos”, y redirecciona al usuario a la página de autenticación.</td></tr>
<tr><td colspan="3">Prototipo de Interfaz</td></tr>
<tr><td>Poscondiciones</td><td colspan="2"></td></tr>
</table>

Tabla 7 Descripción del CUS Gestionar Usuario.

Caso de Uso 2:	Gestionar Usuario.
Actores:	Administrador.
Resumen:	El caso de uso lo inicia el administrador en el momento que necesita asignarles permisos determinados a los usuarios según el rol a desempeñar en el sistema.
Precondiciones:	1. Debe existir un administrador previamente establecido en el sistema.
Referencias	RF 2.
Prioridad	Crítico

Flujo Normal de Eventos

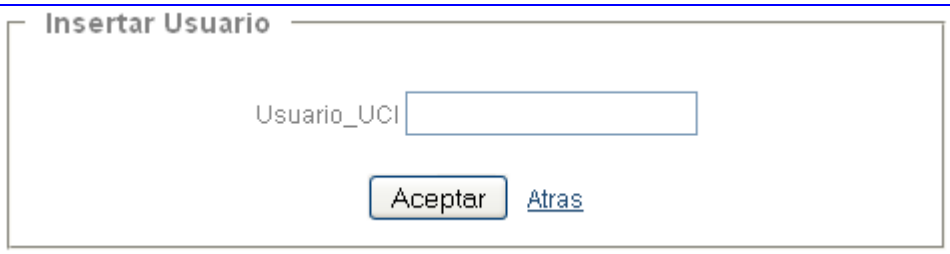
Sección "1"

Acción del Actor	Respuesta del Sistema
1-El administrador que pretende realizar alguna actividad con los usuarios accede al menú de opciones.	2-El sistema muestra el listado de opciones posibles a realizar.
3-El administrador decide: Insertar Usuario (Ver sección Insertar). Eliminar Usuario (Ver sección Eliminar).	

Sección "Insertar"

Acción del Actor	Respuesta del Sistema
1. El administrador desea insertar un nuevo usuario y selecciona la opción "insertar" presente en la interfaz.	2. El sistema muestra una nueva interfaz para llenar los datos del nuevo usuario.
3. El administrador llena los campos de datos y da la opción "Aceptar".	4. El sistema muestra la confirmación:" ¿Está seguro que desea insertar este usuario?"
5. El administrador oprime "Aceptar".	6. El sistema devuelve el mensaje: "El usuario ha sido insertado satisfactoriamente".

Prototipo de Interfaz



Flujos Alternos

Acción del Actor	Respuesta del Sistema

1. Oprimir “Cancelar” cuando se muestra el mensaje de confirmación.	2. No se insertará el nuevo usuario y se redirecciona a la lista de opciones.
3. Insertar los datos sin haber llenado todos los campos previamente.	4. El sistema devuelve el mensaje de error “Debe introducir el nombre del usuario”.
5. Insertar datos incorrectos.	6. El sistema devuelve el mensaje de error: “El usuario es incorrecto”.
7. El administrador oprime “Atrás”.	8. El sistema retorna al listado de opciones de gestión.

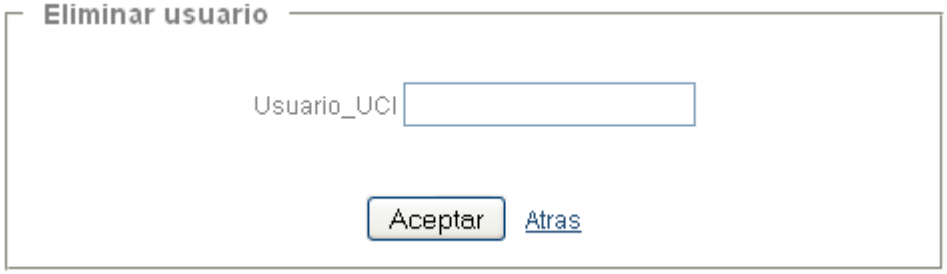
Prototipo de Interfaz

Poscondiciones	

Sección “Eliminar”

Acción del Actor	**Respuesta del Sistema**
5 El administrador desea eliminar un usuario ya existente y selecciona la opción “Eliminar” presente en la interfaz.	6 El sistema muestra el formulario para introducir al usuario que se desea eliminar.
7 El administrador introduce el usuario que se va a eliminar.	8 El sistema muestra la confirmación: ” ¿Está seguro que desea eliminar este usuario?”
9 El administrador oprime “Aceptar”.	10 El sistema devuelve el mensaje: “El usuario ha sido eliminado satisfactoriamente”.

Prototipo de Interfaz



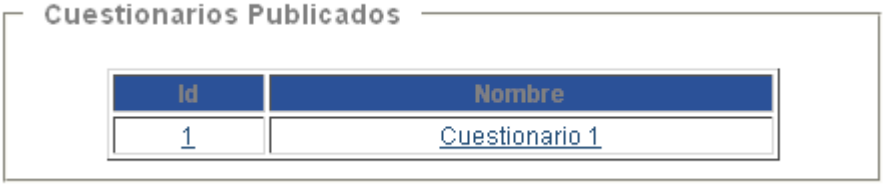
Flujos Alternos

Acción del Actor	**Respuesta del Sistema**

1. Oprimir “Cancelar” cuando se muestra el mensaje de confirmación.	2. No se eliminará el nuevo usuario y se redirecciona a la lista de opciones.
3. Eliminar los datos sin haber llenado todos los campos previamente. Insertar datos incorrectos.	4. El sistema devuelve el mensaje de error “Debe introducir el nombre del usuario”.
5. Eliminar datos incorrectos.	6. El sistema devuelve el mensaje de error: “El usuario es incorrecto”.
7. El administrador oprime “Atrás”.	8. El sistema retorna al listado de opciones de gestión.
Prototipo de Interfaz	
Poscondiciones	

Tabla 8 Descripción del CUS. Realizar Test.

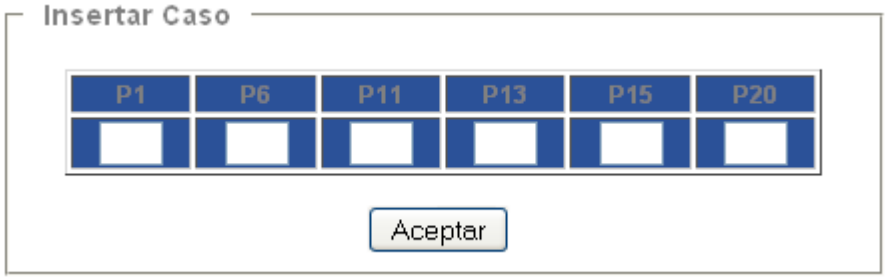
Caso de Uso 5:	Realizar Test
Actores:	Estudiante
Resumen:	El caso de uso se inicia cuando el estudiante va a la aplicación a realizar un test propuesto por el profesor
Precondiciones:	Debe existir el test.
Referencias	RF 5.
Prioridad	Critico
Flujo Normal de Eventos	
Sección “1”	
Acción del Actor	**Respuesta del Sistema**
1. El estudiante escoge la opción realizar cuestionario.	2. El sistema muestra el cuestionario.
3. El estudiante realiza el cuestionario y lo envía	4.
Prototipo de Interfaz	



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	

Tabla 9 Descripción del CUS Gestionar Caso.

Caso de Uso 6:	Gestionar Caso.
Actores:	Profesor.
Resumen:	El caso de uso lo inicia el profesor en el momento que necesita registrar un caso.
Precondiciones:	Debe existir una base de casos.
Referencias	RF 6.
Prioridad	Crítico.
Flujo Normal de Eventos	
Sección "1"	
Acción del Actor	**Respuesta del Sistema**
1. El profesor accede a las opciones de gestión de casos.	2. El sistema muestra las opciones "Insertar" y "Eliminar"
3. El profesor decide: Insertar Caso (Ver sección Insertar). Eliminar Rol (Ver sección Eliminar).	
Sección "Insertar"	
Acción del Actor	**Respuesta del Sistema**
1. El profesor introduce los datos del nuevo caso y pulsa "Aceptar".	2. El sistema muestra el mensaje de confirmación: ¿Está seguro que desea insertar ese caso?
3. El profesor pulsa "Aceptar".	
Prototipo de Interfaz	



Flujos Alternos

Acción del Actor	**Respuesta del Sistema**
1. Cancelar cuando se muestra el mensaje de confirmación	2. El sistema redirecciona al profesor a la página de opciones de gestión.
3. Aceptar dejando campos vacios	4. El sistema muestra el mensaje de error: “No debe dejar campos vacíos”
5. Introducir datos incorrectos	6. El sistema muestra el mensaje de error: “Ha introducido datos incorrectos”

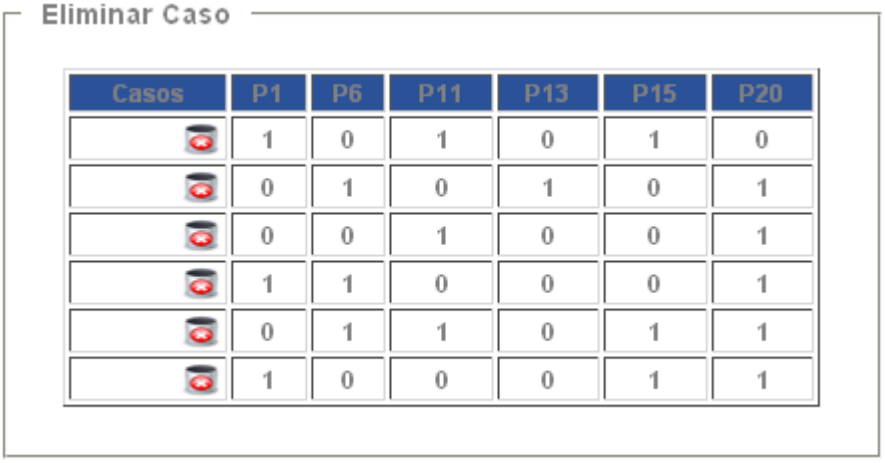
Prototipo de Interfaz

Poscondiciones	

Sección “Eliminar”

Acción del Actor	**Respuesta del Sistema**
1. El profesor escoge el caso a eliminar y pulsa “Eliminar”.	2. El sistema vuelve a la página de opciones de gestión-

Prototipo de Interfaz



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	

Tabla 10 Descripción del CUS Gestionar Material.

Caso de Uso 7:	Gestionar Material.
Actores:	Profesor.
Resumen:	El caso de uso lo inicia el profesor en el momento que necesita realizar alguna acción con los materiales de los estudiantes.
Precondiciones:	
Referencias	RF 7.
Prioridad	Crítico.

Flujo Normal de Eventos	
Sección "1"	
Acción del Actor	**Respuesta del Sistema**
1-El profesor que pretende realizar alguna actividad con los materiales accede al menú de opciones.	2-El sistema muestra las opciones posibles a realizar.
3-El profesor decide: Insertar Material (Ver sección Insertar). Eliminar Material (Ver sección Eliminar).	
Sección "Insertar"	
Acción del Actor	**Respuesta del Sistema**
1-El profesor elige al caso al que desea asignar el material.	2-El sistema muestra la interfaz con el caso elegido.
3-El profesor busca el material en el directorio y pulsa aceptar.	
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	

Sección “Eliminar”	
Acción del Actor	**Respuesta del Sistema**
1-El profesor elige al caso al que desea eliminar el material.	2-El sistema elimina el material.
Prototipo de Interfaz	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
Prototipo de Interfaz	
Poscondiciones	

Tabla 11 Descripción del CUS Gestionar Cuestionario.

Caso de Uso 8:	Gestionar Cuestionario.
Actores:	Profesor.
Resumen:	El caso de uso lo inicia el profesor en el momento que necesita subir un cuestionario al sistema o realizar acciones sobre los existentes.
Precondiciones:	
Referencias	RF 8.
Prioridad	Crítico.
Flujo Normal de Eventos	
Sección “1”	
Acción del Actor	**Respuesta del Sistema**
1. El profesor que pretende realizar alguna actividad con los cuestionarios accede al menú de opciones.	2. El sistema muestra el listado de opciones posibles a realizar.
3. El administrador decide: Asignar Rol (Ver sección Insertar). Eliminar Rol (Ver sección Eliminar).	
Sección “Insertar”	
Acción del Actor	**Respuesta del Sistema**
1. El profesor desea insertar un nuevo cuestionario y selecciona la opción “Insertar” presente en la interfaz.	2. El sistema muestra una nueva interfaz para llenar los datos del nuevo cuestionario.

3. El profesor llena los campos de datos y da la opción “Aceptar”.	4. El sistema muestra la confirmación:” ¿Está seguro que desea insertar el cuestionario?
5. El profesor oprime “Aceptar”.	6. El sistema devuelve el mensaje: “El cuestionario ha sido insertado satisfactoriamente”.

Prototipo de Interfaz

Flujos Alternos

Acción del Actor	Respuesta del Sistema
1. Oprimir “Cancelar” cuando se muestra el mensaje de confirmación.	2. No se inserta el nuevo cuestionario y se redirecciona a la pagina del formulario de entrada de datos.
3. Insertar cuestionario sin haber llenado el campo nombre.	4. El sistema devuelve el mensaje de error “Debe introducir el nombre del cuestionario”.
5. El administrador oprime “Ver”.	6. El sistema muestra la lista de cuestionarios.

Prototipo de Interfaz

Poscondiciones	

Sección “Eliminar”

Acción del Actor	Respuesta del Sistema
1. El profesor desea eliminar un cuestionario y selecciona la opción “Eliminar” presente en la interfaz.	2. El sistema muestra una nueva interfaz para insertar el id del cuestionario a eliminar.
3. El profesor llena los campos de datos y da la opción “Aceptar”.	4. El sistema muestra la confirmación:” ¿Está seguro que desea eliminar el cuestionario?

5. El profesor oprime “Aceptar”.	6. El sistema devuelve el mensaje: “El cuestionario ha sido eliminado satisfactoriamente”.
Prototipo de Interfaz	
	
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. Oprimir “Cancelar” cuando se muestra el mensaje de confirmación.	2. No se elimina el y se redirecciona a la página del formulario de entrada de datos.
3. Eliminar cuestionario sin introducir el id.	4. El sistema devuelve el mensaje de error “Debe introducir el id del cuestionario”.
5. Introducir id incorrecto	6. El sistema muestra el mensaje “El id del cuestionario es incorrecto”.
7. El administrador oprime “Ver”.	8. El sistema vuelve a la página anterior.
Prototipo de Interfaz	
Poscondiciones	

3.5 Propuesta de Arquitectura del sistema

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad y disponibilidad. (Frómeta, 1995)

Las estructuras a las que se refieren en este concepto son las llamadas estáticas y dinámicas, o sea las representaciones estáticas de los objetos del dominio del software en general así como la relación entre ellos con su entorno.

Es necesario tener en cuenta que los datos pueden crecer exponencialmente a medida que aparezcan nuevos tipos de estudiantes no registrados en la Base de Casos. Además de que en el futuro se pueden elevar los niveles de concurrencia si se pretende por ejemplo que el estudiante pueda estar realizando el test mientras el profesor actualiza los documentos. Esto no debe en ningún momento afectar el rendimiento o tiempo de repuesta del software.

3.5.1 Descripción de la estructura de la solución propuesta.

Como se ha ilustrado hasta este momento, un Sistema de Enseñanza Aprendizaje Inteligente cuenta con los módulos: Interfaz, Estudiante, Tutor y Pedagógico. Para dar solución al problema que supone la construcción de este, se optó por la utilización del razonamiento basado en casos que es un tipo de Sistema Basado en Conocimientos. A partir de ello se reorganizan los módulos de la siguiente manera: Interfaz, Base de Casos (rasgos predictores), Adaptación y Recuperación que serían los rasgos objetivos que engloban los anteriores módulos Tutor y Pedagógico. Hasta aquí está demostrado que se necesita desde el punto de vista ingenieril una arquitectura que sea capaz de separar las acciones de cada uno de las partes componentes del sistema sin ocuparse aún por las interacciones entre ellas. De modo que las opciones a considerar, teniendo en cuenta la experiencia de los desarrolladores como elemento adicional, son: Arquitectura en capas y Patrón Modelo Vista Controlador.

En una aplicación con una arquitectura de capas, cada capa se encarga de una tarea determinada y una capa solo puede utilizar la capa inferior a ella y ni siquiera conocerá las capas superiores. (Medín, 2008)

La anterior estructura de tres capas tiene sus limitaciones. Si en un momento determinado se desea recuperar del por ejemplo un listado de estudiantes dado un grupo, ¿desde dónde se invocaría este método, o a cualquier otro método que invocase a dicho método?

Pues desde la interfaz. Pero de ese modo, en el que es la propia interfaz la responsable de invocar los métodos del negocio, se está haciendo a la interfaz completamente dependiente del negocio y esto viola los principios que siguen tanto los SEAI como los SBC.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y

las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. (Potencier, 2008)

Por tanto se puede concluir que la mejor opción es utilizar el patrón Modelo Vista Controlador en la modificación de este que más conveniente sea.

3.5.2 Descripción de una arquitectura que sustente la solución.

Modelo Vista Controlador como estilo arquitectónico, tiene entre sus variantes la que se muestra en la figura 4.

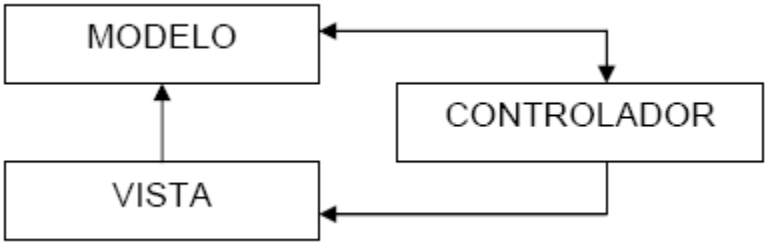


Figura 10. Variante Intermedia del Patrón Modelo Vista Controlador.

En esta el controlador es el encargado de ejecutar todo el flujo de acciones y envío de datos a la vista y desde modelo. Aunque independientemente de toda variante el ciclo de control es el siguiente:

- El usuario interactúa con la interfaz a través del mouse o el teclado, pulsando un botón o enlace.
- El controlador recibe la petición y gestiona el evento que llega.
- El controlador accede al modelo actualizándolo de acuerdo a la petición realizada.
- El controlador deleja a la vista la responsabilidad de mostrar la interfaz de usuario con los datos solicitados
- La interfaz espera nuevas acciones reiniciando el ciclo nuevamente.

De manera que a partir de lo planteado, el modelo es todo lo concerniente a los datos de la Base de Casos y la Base de Datos, el controlador incluye los módulos de Adaptación y Recuperación y la vista se encarga de la interfaz.

3.5.2.1 Modelo Vista Controlador en Symfony 1.2.9

La capa del modelo es la más compleja del framework Symfony. Una de las razones de esta complejidad es que la manipulación de datos es una tarea bastante intrincada. Las consideraciones de seguridad relacionadas con el modelo son cruciales para un sitio web y no deberían ignorarse. Otra de las razones es que Symfony se ajusta mejor a las aplicaciones medianas y grandes en un entorno empresarial. En ese tipo de aplicaciones, las tareas automáticas proporcionadas por el

modelo de Symfony suponen un gran ahorro de tiempo, por lo que merece la pena el tiempo dedicado a aprender su funcionamiento interno. (Potencier, 2008)

En el caso del que se ocupa este trabajo el proyecto no es de gran alcance pero el dominio del modelo es determinante para el funcionamiento correcto del sistema dado que el mismo depende de la base inicial de casos y se nutre del aumento de los mismos.

La vista se encarga de producir las páginas que se muestran como resultado de las acciones. La vista en Symfony está compuesta por diversas partes, estando cada una de ellas especialmente preparada para que pueda ser fácilmente modificable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones. Independientemente del tipo de trabajo, existen herramientas y utilidades para simplificar y acelerar el trabajo (normalmente tedioso) de presentar los resultados de las acciones. (Potencier, 2008)

La vista no es la interfaz como algunos erróneamente piensan, de hecho la vista al igual que el controlador y el modelo trabajan en el lado del servidor. Los objetos de la vista se encargan de generar las páginas que se muestran en la interfaz.

En Symfony, la capa del controlador está dividida en dos partes: el controlador frontal, que es el único punto de entrada a la aplicación para un entorno dado, y las acciones, que contienen la lógica de las páginas. El controlador frontal es el único punto de entrada a la aplicación. Carga la configuración y determina la acción a ejecutarse. Las acciones contienen la lógica de la aplicación. Verifican la integridad de las peticiones y preparan los datos requeridos por la capa de presentación. (Potencier, 2008)

En resumen, la versión de symfony escogida es la que más se corresponde con la arquitectura que se necesita para desarrollar el sistema propuesto.

3.6 Conclusiones

En este capítulo se trata la descripción del sistema propuesto, para ello primeramente se desarrolla un modelo de dominio para un mejor entendimiento de los conceptos relacionados con la solución y el esclarecimiento de las funcionalidades que esta debe satisfacer. Tomando esto como base se lleva a cabo un sistema de modelación en términos de casos de usos que una vez especificados garantizan una compresión exacta de lo que se desea desarrollar. Todos los artefactos generados sirven de entrada para la etapa posterior, el diseño correspondiente a la disciplina de modelado que se desarrolla durante la fase de construcción según la metodología que se está usando como guía para el trabajo. Además se propone la arquitectura del sistema que está basada en el patrón Modelo Vista Controlador (MVC).

CAPÍTULO 4

Construcción de la solución propuesta.

4.1 Introducción

En este capítulo se aborda todo lo referente al diseño del sistema. Se modela el sistema y se le proporciona forma, para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. También se describe todo lo referente a la implementación del sistema. Además se muestran los diagramas que corresponden al flujo de implementación, como es el caso del diagrama de despliegue que muestra mediante nodos como están distribuidos los recursos que intervienen e interactúan con el sistema y el diagrama de componentes que muestra la relación física entre estos.

4.2 Diseño del sistema

En este epígrafe se describe el diseño del software adaptativo propuesto para apoyar el proceso de enseñanza-aprendizaje de Matemática I en el curso introductorio de Matemática I, en la Ingeniería en Ciencias Informáticas de la UCI. El desarrollo del SEAI que se obtiene como resultado de este trabajo de este diseño está a disposición de todos los profesores, con el fin de apoyarlos en su actividad docente; y de todos los alumnos para que lo utilicen como una herramienta interactiva-adaptativa de apoyo a su aprendizaje individual. El diseño es modular y flexible, pues está basado en los módulos propuestos por los sistemas de razonamiento basados en casos explicados en el epígrafe 1.4.2.

4 2.1 Patrones de Diseño.

Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados. (Medín, 2008).

Entre los patrones de diseño se encuentra los conocidos como patrones GRASP (Patrones Generales de Software para Asignar Responsabilidades): Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Los primeros cinco patrones GRASP son:

- **Experto**: se encarga de asignar una responsabilidad al experto en información, o sea, la clase con la información suficiente para cubrir la necesidad.
- **Creador**: se encarga de darle a la clase A la responsabilidad de crear objetos de la clase B. En este caso A es creador de los objetos B.
- **Alta Cohesión:** se encarga de asignar responsabilidades a las clases de modo que las funcionalidades estén estrechamente relacionadas y que cada elemento del diseño tenga una labor única. Una clase con baja cohesión se hace difícil de entender, de conservar y de reutilizar.
- **Bajo Acoplamiento:** garantiza que una clase no dependa de muchas otras ya que se hace difícil de reutilizar.
- **Controlador**: propone asignarle la responsabilidad de manejar todos los eventos del sistema a una clase. (Medín, 2008)

Por otro lado están los patrones GOF (Gang of Four, en inglés) que según su propósito se clasifican en:

- **Patrones de creación:** presentan la guía de cómo crear objetos cuando sus creaciones requieren tomar decisiones, las cuales serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos delegar responsabilidades.
- **Patrones estructurales:** describen la forma en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros.
- **Patrones de comportamiento:** se utilizan para organizar, manejar y combinar comportamientos. (Hernández, 2002)

4.2.1.1 Aplicación de los Patrones

- **Patrón Creador:** En Symfony la capa del controlador contiene el código que une la lógica del negocio con la presentación. Este cuenta con varios componentes entre los que se identifican las acciones que son los métodos que, utilizan el modelo y definen variables para la vista. En la aplicación en cuestión se cuenta con un archivo de tipo actions.class.php por cada módulo. Estas acciones se ejecutan en las clases administracionActions, cuestionarioActions y loginActions de los módulos administración, cuestionario y login respectivamente. En las actions mencionadas se crean los objetos que representan las entidades evidenciando que son “creadoras” de dichas clases.

- **Patrón Experto:** Es uno de los patrones que se emplean al trabajar con el Framework Symfony, se evidencia en la inclusión de **Propel** para mapear la base de datos. Este último, genera las clases para la gestión de la información de las tablas de dicha base de datos con las responsabilidades debidamente asignadas según el patrón Experto. Cada una de estas clases cuenta con un conjunto de funcionalidades que las convierte en expertas de la información de la tabla a la que representa. Además la aplicación de este patrón permite mover parte de la lógica del negocio hacia las clases modelo evitado así la sobrecarga del controlador.
- **Patrón Alta Cohesión:** Una de las características principales del Framework Symfony es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, la clase administracionActions contiene responsabilidades estrechamente relacionadas, encargadas de controlar las acciones de las plantillas. Esto hace posible que el software sea flexible a cambios sustanciales con efecto mínimo.
- **Patrón Bajo Acoplamiento:** En el modelo es donde se pueden encontrar relaciones de herencia y asociación entre las clases, pero no es de gran jerarquía.
- **Patrón Controlador:** Las clases administracionActions**,** cuestionarioActions y loginActions son las controladoras del sistema y su función es servir de enlace entre la vista y el modelo por módulos. Precisamente es esta la evidencia de la existencia del patrón Controlador en el proyecto, aunque es común que este se implemente asignando el control a una sola clase, de este modo se asegura que no se sobrecargue el controlador. Además, Symfony implementa además el patrón Front Controller (Controlador Frontal), en el cual existen varias clases controladoras que forman un flujo para atender las peticiones del usuario y de otras clases.

Creacionales:

- Singleton (Instancia única)

  El patrón Singleton garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En el controlador frontal hay una llamada a sfContext::createInstance($configuration)->dispatch(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de symfony.
- Abstract Factory (Fábrica abstracta)

  El patrón Abstract Factory permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una

petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Estructurales:

- Decorator (Envoltorio)

  El patrón Decorator añade funcionalidad a una clase dinámicamente. El archivo layont.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado.

Comportamiento:

- Observer (Observador)

  El patrón Observer define una dependencia del tipo uno-a-muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, el observador se encarga de notificar este cambio a todos los otros dependientes. El objetivo de este patrón es desacoplar la clase de los objetos clientes del objeto, aumentando la modularidad del lenguaje.

  Symfony proporciona un sistema de eventos inspirado en el patrón Observer para suplir las limitaciones actuales de PHP de que una clase no puede heredar de más de una clase.

4.2.2 Vista Lógica del diseño

Describe las partes arquitectónicamente significativas del modelo de diseño, como son la descomposición en capas, subsistemas o paquetes.

La raíz de cualquier proyecto en symfony tiene la estructura siguiente:

- apps
- cache
- config
- data
- doc
- lib
- log
- plugins
- test
- web

En el diagrama que se presenta a continuación sólo se tienen en cuenta los directorios apps, web y lib ya que son relevantes para la visualización del patrón arquitectónico planteado aplicado al desarrollo del proyecto.

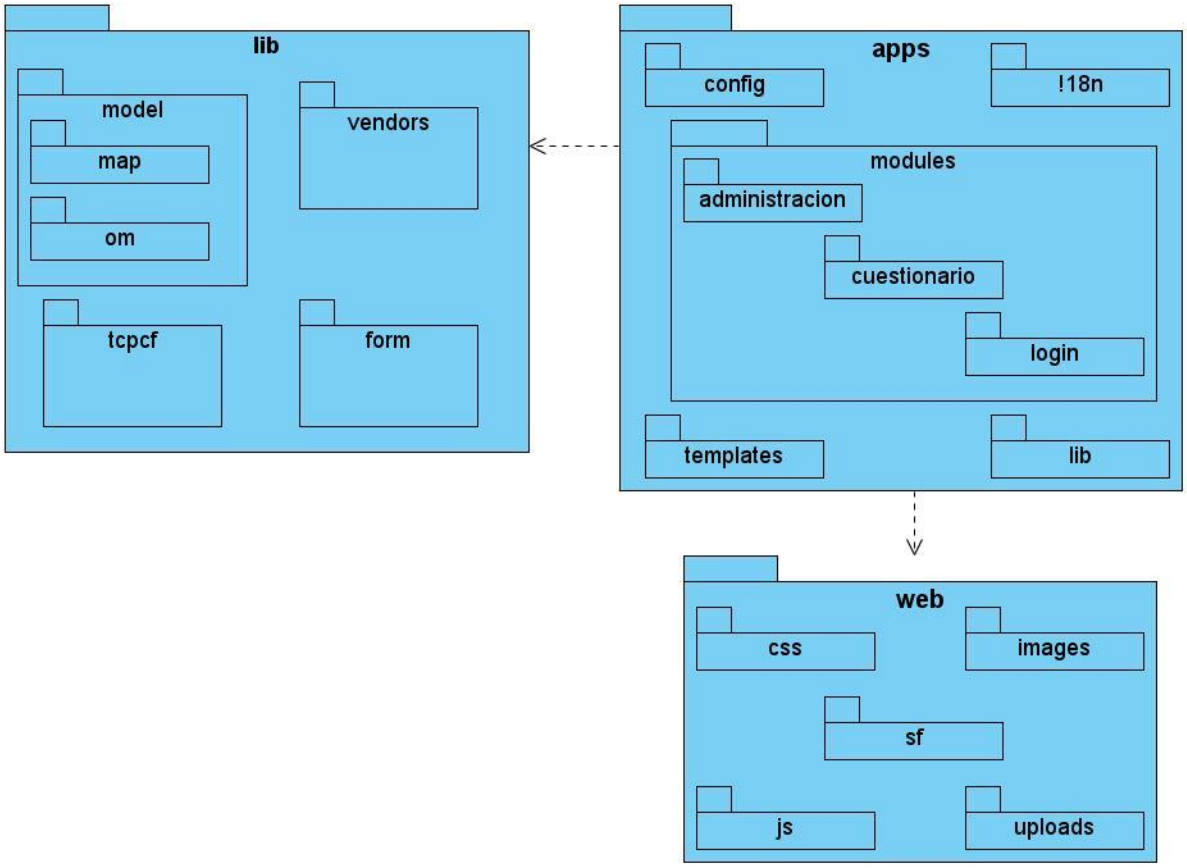


Figura 11. Vista Lógica de la Arquitectura.

4.2.3 Diseño de clases.

Como se mencionó anteriormente se aplicó el patrón Modelo-Vista-Controlador. El paquete Vista agrupa todas las clases de la presentación como las páginas servidoras, que son las encargadas de construir las páginas clientes, las que a su vez están compuestas por un formulario, aunque no todas.

En el paquete Modelo se encuentra la capa de acceso a datos. Sólo se representan cuatro clases que sintetizan las relaciones principales existentes entre ellas para minimizar la complejidad del diagrama. En el paquete Controlador están el Controlador frontal index.php que se encarga de redireccionar todas las peticiones que se hacen desde de la vista y las clases "administracionActions", "cuestionarioActions" y "loginActions" controlan todo el proceso y además de establecen la comunicación entre la vista y el modelo. También se observan el subsistema Propel que genera la capa de acceso a datos y el subsistema Componentes de Symfony que agrupa todas las clases y funcionalidades del framework Symfony. (Ver anexos del 1 al 6)

4.2.4 Diseño de la Base de Datos

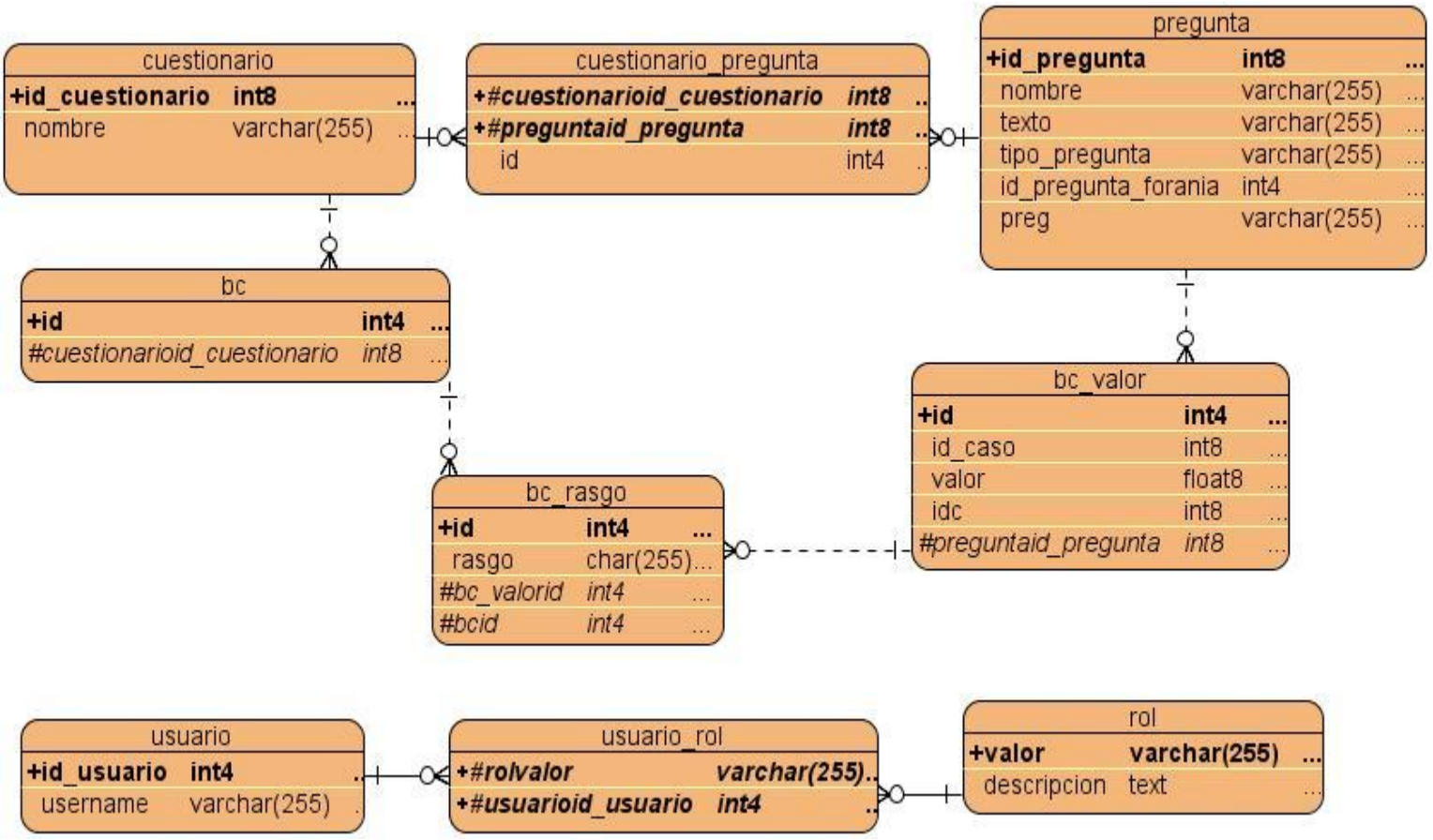


Figura 12. Modelo Físico de la Base de Datos.

4.3 Generalidades de la Implementación

4.3.1 Diagrama de Despliegue

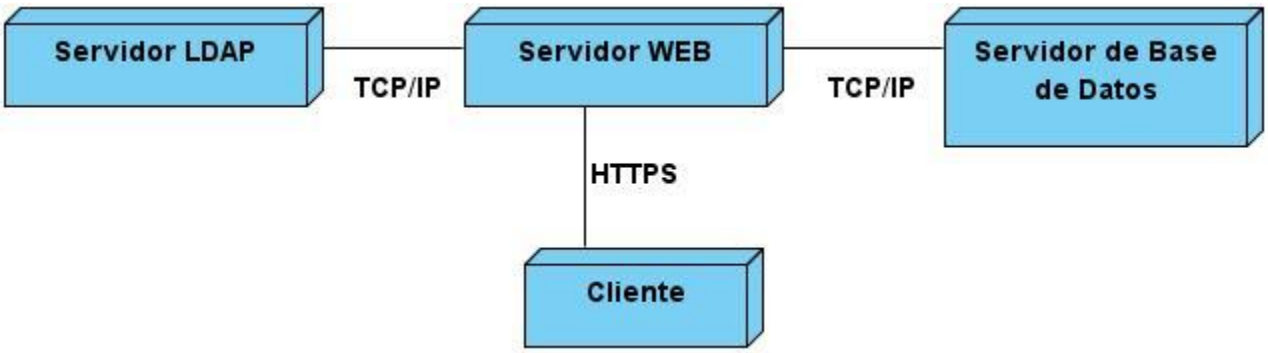


Figura 13. Diagrama de despliegue.

La vista de despliegue permite ver el sistema en términos de nodos de procesamiento, servidores o dispositivos. Muestra la comunicación entre los diferentes nodos que componen los escenarios de distribución física del sistema. Los diagramas de despliegue muestran la configuración en funcionamiento del sistema, incluyendo su hardware y su software.

En el caso del sistema que se quiere implementar se tiene en cuenta que se desarrolla como una aplicación web, en la cual los componentes que la conforman se encuentran de manera

distribuida como se representa en el diagrama de despliegue mostrado en la figura n. Está presente la computadora del cliente desde donde se accede a la aplicación a través de un navegador, esta se encuentra en un servidor web que a su vez se conecta al servidor de base de datos y al Directorio Activo de la Universidad para el proceso de autenticación de los usuarios.

4.3.2 Diagrama de Componentes

Un diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes software, sean éstos componentes de código fuente, binarios o ejecutables. Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. (Fernández, 2001)

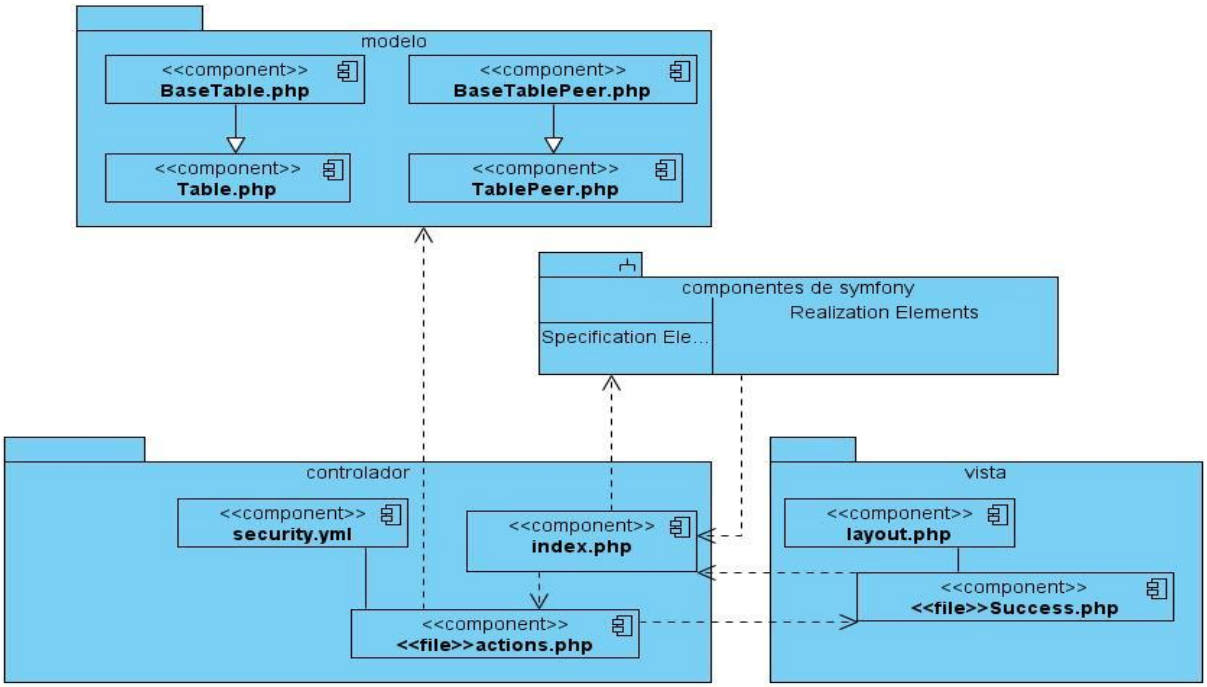


Figura 14. Diagrama de Componentes.

Symfony es un framework basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por los tres niveles mencionados en el epígrafe 3.5.2.1:

- El Modelo representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La Vista transforma el modelo en una página web que permite al usuario interactuar con ella.
- El Controlador se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Los componentes básicos de la aplicación que se realiza están distribuidos del modo siguiente:

La capa del Modelo

Abstracción de la base de datos.

Acceso a los datos.

La capa de la Vista

Plantilla.

Layout.

La capa del Controlador

Controlador frontal.

Acción.

Todas las peticiones web son manejadas por un solo controlador frontal que es el punto de entrada único de toda la aplicación en un entorno determinado. En el diagrama está representado el index.php que es el que controla el entorno de producción. El entorno de desarrollo lo maneja el SEAI.php. Cuando el controlador frontal recibe una petición, utiliza el sistema de enrutamiento para asociar el nombre de una acción y el nombre de un módulo con la dirección escrita por el usuario.

Las acciones utilizan el modelo y definen variables para la vista. Cuando se realiza una petición en la aplicación, la URL define una acción y los parámetros de la petición. Para restringir el acceso a las acciones se crea y se edita un archivo de configuración YAML llamado security.yml en el directorio config/ del módulo administración.

La capa de la vista también puede aprovechar la separación de código. Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla. El layout es global en toda la aplicación.

La plantilla sólo se encarga de visualizar las variables definidas en el controlador. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este comportamiento es una implementación del patrón de diseño llamado "decorator”.

Las clases de objetos que están en el directorio lib/model heredan de las clases con nombre Base. Estas clases no se modifican cuando se ejecuta la tarea propel:build-model, por lo que son las clases en las que se añaden los métodos. Además son clases objeto que representan un registro de la base de datos. Permiten acceder a las columnas de un registro y a los registros relacionados. Por tanto, es posible obtener el título de un artículo invocando un método del objeto Table.

Las clases con nombre Base del directorio lib/model/om/ son las que se generan directamente a partir del esquema. Nunca se deberían modificar esas clases, porque cada vez que se genera el modelo, se borran todas las clases.

Existe una quinta clase que se crea en el directorio lib/model/map/ y que contiene metainformación relativa a la tabla que es necesaria para la ejecución de la aplicación. Pero como es una clase que nunca se modifica no se considera necesario incluirla en el diagrama y se considera parte del subsistema representado por el resto de los componentes de symfony.

Las clases de tipo peer son clases que tienen métodos estáticos para trabajar con las tablas de la base de datos. Proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase relacionada.

4.4 Conclusiones

En el transcurso de este capítulo se obtuvieron los artefactos más significativos del diseño de la aplicación, estos sirven de entrada para el flujo de implementación, facilitando un mayor entendimiento de la aplicación a los desarrolladores, como son los diagramas de clases del diseño, para lograr un mayor entendimiento de las funcionalidades a implementar, el diagrama Entidad Relación de la base de datos, así como también la utilidad de los diferentes patrones de diseño en el desarrollo de la aplicación.

Se ha mostrado la información referente a la etapa de implementación, empezando por el diagrama de despliegue, donde se detalló la relación hardware y software del sistema, además del diagrama de componentes. Todos los elementos mencionados anteriormente permitieron la construcción de un sistema eficiente y con mayor calidad.

CAPÍTULO 5

Prueba del sistema desarrollado.

5.1 Pruebas.

Dado que SEAI es un software producto de un proceso ingenieril, del cual se conocen las funcionalidades específicas para las cuales fue diseñado, se pueden llevar a cabo pruebas que demuestran que cada función es completamente operativa. Este enfoque se refiere a la pruebas de caja negra que se llevan a cabo sobre la interfaz de la aplicación. Por esta razón los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Estas pruebas examinas algunos aspectos del sistema sin tener mucho en cuenta la estructura interna del software.

5.2 Objetivo

El objetivo de realizar este tipo de prueba al sistema, es detectar el incorrecto o incompleto funcionamiento de este, así como los errores de interfaces, rendimiento, errores de inicialización y terminación.

5.3 Alcance

El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar el comportamiento de la interfaz gráfica, su interacción con el usuario y la calidad funcional.

5.4 Descripción

Dentro del método de Caja Negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar. El método que se propone para diseñar los casos de pruebas es una variante del de particiones equivalentes, propuesto por RUP y al que se le han

añadido algunas consideraciones, este método consta de 3 pasos fundamentales y usa como artefacto de entrada los casos de usos.

5.5 Casos de Prueba

Para verificar que se cumplieran los requerimientos funcionales establecidos, se realizó la prueba a los casos de uso más críticos. Esta se hace con el objetivo de evaluar la interacción del usuario con el sistema, dependiendo de las funcionalidades a las que tenga acceso el usuario y además se verificó que cada una de estas se correspondiera, con las descripciones de cada uno de los casos de uso del sistema realizadas.

5.5.1 Sistema completo de escenarios por casos de uso.

Tabla 12 Matriz parcial de escenarios. CUS Autenticar Usuario.

Nombre del escenario	Flujo donde empieza	Flujo Alterno
Escenario 1: “Introducir usuario y contraseña válidos”.	Flujo básico de los eventos.	
Escenario 2: “Introducir datos incorrectos”.		Flujo alterno 1.
Escenario 3: “Introducir datos vacios”.		Flujo alterno 2.

Tabla 13 Matriz parcial de escenarios. CUS Gestionar Usuario.

Nombre de la sección	Nombre del escenario	Flujo donde empieza	Flujo Alterno
“Insertar”	Escenario 1: “Introducir usuario válido”.	Flujo básico de los eventos.	
	Escenario 2: “Cancelar acción”.		Flujo alterno 1.
	Escenario 3: “Introducir usuario vacío”.		Flujo alterno 2.
	Escenario 4: “Introducir datos incorrectos”.		Flujo alterno 3.
	Escenario 5: “Volver atrás”.		Flujo alterno 4.
“Eliminar”	Escenario 1: “Introducir usuario válido”.	Flujo básico de los eventos.	
	Escenario 2: “Cancelar		Flujo alterno 1.

	acción”.		
	Escenario 3: “Introducir usuario vacío”.		Flujo alterno 2.
	Escenario 4: “Introducir datos incorrectos”.		Flujo alterno 3.
	Escenario 5: “Volver atrás”.		Flujo alterno 4.

Tabla 14 Matriz parcial de escenarios. CUS Realizar Test.

Nombre del escenario	Flujo donde empieza	Flujo Alterno
Escenario 1: “Responder test”	Flujo básico de los eventos	
Escenario 2: “Enviar test”.	Flujo básico de los eventos	

Tabla 15 Matriz parcial de escenarios. CUS Gestionar Caso.

Nombre de la sección	Nombre del escenario	Flujo donde empieza	Flujo Alterno
“Insertar”	Escenario 1: “Introducir caso válido”.	Flujo básico de los eventos.	
	Escenario 2: “Cancelar acción”.		Flujo alterno 1.
	Escenario 3: “Introducir caso vacío”.		Flujo alterno 2.
	Escenario 4: “Introducir datos incorrectos”.		Flujo alterno 3.
“Eliminar”	Escenario 1: “Eliminar caso seleccionado”.	Flujo básico de los eventos.	

Tabla 16 Matriz parcial de escenarios. CUS Gestionar Material.

Nombre de la sección	Nombre del escenario	Flujo donde empieza	Flujo Alterno
“Insertar”	Escenario 1: “Elegir caso a insertarle material”.	Flujo básico de los eventos.	
	Escenario 2: “Buscar material en el directorio”.	Flujo básico de los eventos.	
“Eliminar”	Escenario 1: “Elegir caso a eliminarle material”.	Flujo básico de los eventos.	

Tabla 17 Matriz parcial de escenarios. CUS Gestionar Cuestionario.

Nombre de la sección	Nombre del escenario	Flujo donde empieza	Flujo Alterno
"Insertar"	Escenario 1: "Introducir nombre del cuestionario".	Flujo básico de los eventos.	
	Escenario 2: "Cancelar acción".		Flujo alterno 1.
	Escenario 3: "Introducir nombre vacío".		Flujo alterno 2.
	Escenario 4: "Ver listado actualizado".		Flujo alterno 3.
"Eliminar"	Escenario 1: "Introducir id válido".	Flujo básico de los eventos.	
	Escenario 2: "Cancelar acción".		Flujo alterno 1.
	Escenario 3: "Introducir id vacío".		Flujo alterno 2.
	Escenario 4: "Introducir id incorrecto".		Flujo alterno 3.
	Escenario 5: "Ver listado actualizado".		Flujo alterno 4.

5.5.2 Casos de Prueba con los valores de datos.

Tabla 18 Matriz de Casos de pruebas con los valores de los datos. CUS Autenticar Usuario.

Escenario	**"usuario"**	**"contraseña"**	**Respuesta**
Escenario 1: "Introducir usuario y contraseña válidos".	**lpazos**	**supernatural**	El sistema brinda acceso a las diferentes vistas de la aplicación en dependencia de los derechos del usuario definidos por su rol
Escenario 2: "Introducir datos incorrectos".	**5874585**		Se muestra un mensaje de error en los datos: "Usuario o contraseña incorrectos", y redirecciona al usuario a la página de autenticación.
Escenario 3: "Introducir datos vacios".			El sistema muestra el mensaje "Usuario o contraseña incorrectos",

			y redirecciona al usuario a la página de autenticación.

Tabla 19 Matriz de Casos de pruebas con los valores de los datos. CUS Gestionar Usuario.

Sección	Escenario	"usuario"	Respuesta
"Insertar"	Escenario 1: "Introducir usuario válido".	**dmunoz**	El sistema muestra la confirmación:" ¿Está seguro que desea insertar este usuario?" y devuelve el mensaje: "El usuario ha sido insertado satisfactoriamente".
	Escenario 2: "Cancelar acción".		No se insertará el nuevo usuario y se redirecciona a la lista de opciones.
	Escenario 3: "Introducir usuario vacío".		El sistema devuelve el mensaje de error "Debe introducir el nombre del usuario".
	Escenario 4: "Introducir datos incorrectos".	**35873458**	El sistema devuelve el mensaje de error: "El usuario es incorrecto".
	Escenario 5: "Volver atrás".		El sistema retorna al listado de opciones de gestión.
"Eliminar"	Escenario 1: "Introducir usuario válido".	**dmunoz**	El sistema muestra la confirmación:" ¿Está seguro que desea eliminar este usuario?" y devuelve el mensaje: "El usuario ha sido eliminado satisfactoriamente".
	Escenario 2: "Cancelar acción".		No se eliminará el nuevo usuario y se redirecciona a la lista de opciones.

	Escenario 3: “Introducir usuario vacío”.		El sistema devuelve el mensaje de error “Debe introducir el nombre del usuario”.
	Escenario 4: “Introducir datos incorrectos”.	**465776**	El sistema devuelve el mensaje de error: “El usuario es incorrecto”.
	Escenario 5: “Volver atrás”.		El sistema retorna al listado de opciones de gestión.

Tabla 20 Matriz de Casos de pruebas con los valores de los datos. CUS Realizar Test.

Escenario	**“cuestionario”**	**Respuesta**
Escenario 1: “Responder test”	**Cuestionario 1**	El sistema muestra el cuestionario.
Escenario 2: “Enviar test”.		El sistema muestra al estudiante el material por el que debe estudiar.

Tabla 21 Matriz de Casos de pruebas con los valores de los datos. CUS Gestionar Caso.

Sección	**Escenario**	**“caso 1”**	**“caso n”**	**Respuesta**
“Insertar”	Escenario 1: “Introducir caso válido”.	**1**	**0**	El sistema muestra el mensaje de confirmación: ¿Está seguro que desea insertar ese caso? Luego inserta el nuevo caso.
	Escenario 2: “Cancelar acción”.			El sistema redirecciona al profesor a la página de opciones de gestión.
	Escenario 3: “Introducir caso vacío”.			El sistema muestra el mensaje de error: “No debe dejar campos vacíos”
	Escenario 4: “Introducir datos incorrectos”.	**5**	**3**	El sistema muestra el mensaje de error: “Ha introducido datos incorrectos”
“Eliminar”	Escenario 1:			El sistema elimina el caso y

	"Eliminar caso seleccionado".			vuelve a la página de opciones de gestión.

Tabla 22 Matriz de Casos de pruebas con los valores de los datos. CUS Gestionar Material.

Sección	Escenario	"caso"	Respuesta
"Insertar"	Escenario 1: "Elegir caso a insertarle material".	**Caso1**	
	Escenario 2: "Buscar material en el directorio".		
"Eliminar"	Escenario 1: "Elegir caso a eliminarle material".	**Caso1**	

Tabla 23 Matriz de Casos de pruebas con los valores de los datos. CUS Gestionar Cuestionario.

Sección	Escenario	"nombre"	"id"	Respuesta
"Insertar"	Escenario 1: "Introducir nombre del cuestionario".	**Cuestionario 2**		El sistema muestra la confirmación:" ¿Está seguro que desea insertar el cuestionario? Luego inserta el cuestionario
	Escenario 2: "Cancelar acción".			No se inserta el nuevo cuestionario y se redirecciona a la pagina del formulario de entrada de datos.
	Escenario 3: "Introducir nombre vacío".			El sistema devuelve el mensaje de error "Debe introducir el nombre del cuestionario".

	Escenario 4: “Ver listado actualizado”.			El sistema muestra la lista de cuestionarios.
“Eliminar”	Escenario 1: “Introducir id válido”.		**2**	El sistema elimina el cuestionario y devuelve el mensaje: “El cuestionario ha sido eliminado satisfactoriamente”.
	Escenario 2: “Cancelar acción”.			No se elimina el y se redirecciona a la pagina del formulario de entrada de datos.
	Escenario 3: “Introducir id vacío”.			El sistema devuelve el mensaje de error “Debe introducir el id del cuestionario”.
	Escenario 4: “Introducir id incorrecto”.		**300**	El sistema muestra el mensaje “El id del cuestionario es incorrecto”.
	Escenario 5: “Ver listado actualizado”.			El sistema vuelve a la página anterior.

5.5.3 Resumen de Resultados.

Tabla 24 Resumen de Pruebas.

Caso de uso	Respuesta del sistema	Resultado
Autenticar Usuario	Si el usuario y la contraseña	Satisfactorio

	son correcto el sistema autentica al usuario, de lo contrario lo redirecciona a la página de autenticación.	
Gestionar Usuario	Si el usuario es correcto es insertado de lo contrario se muestra un mensaje de error.	Satisfactorio
Realizar Test	El sistema muestra los cuestionarios disponibles y muestra los materiales cuando se envía el test.	Satisfactorio
Gestionar Caso	El sistema muestra el formulario de entrada de datos en dependencia de la cantidad de preguntas del cuestionario.	Satisfactorio
Gestionar Material	El sistema brinda la posibilidad de insertar y eliminar materiales por casos.	Satisfactorio
Gestionar Cuestionario	El cuestionario se inserta correctamente aunque es recomendable que tenga nombre y se elimina sólo si el id es bien insertado.	Satisfactorio

5.6 Conclusiones.

Con la realización de las pruebas descritas se ha podido comprobar que el sistema es funcional y se encuentra correctamente validado cumpliendo así con los requerimientos especificado por el cliente y en correspondencia con las descripciones de los casos de uso del sistema realizadas.

Después de haber realizado un profundo estudio acerca de los procedimientos que conforman el proceso de evaluación de los estudiantes en la asignatura Matemática I; el estado del arte de los SEAI y el RBC como estrategia para implementar los primeros; además de pasar por las distintas etapas que lleva la construcción de un software, como son Negocio, Requerimientos, Diseño e Implementación se arriba a las siguientes conclusiones:

- El SEAI de apoyo al curso introductorio de matemática I es un prototipo que cumple con especificaciones importantes para un sistema tutorial inteligente. El sistema funciona como sistema de consulta y como sistema tutorial.
- La estructura modular del STI, permite que su adaptación a otros dominios de aplicación sea más fácil, como en este caso, pues únicamente se debe cambiar la materia de los cuestionarios.
- El sistema presentado es un intento por tener un sistema tutorial que permita que el estudiante este comprometido y motivado a aprender.
- El SEAI constituye un avance importante hacia un sistema tutorial inteligente enfocado a la capacitación personalizada en la UCI.

Tomando como punto de partida los resultados obtenidos con la realización de este trabajo de diploma y el cumplimiento del Objetivo General, se recomienda:

- El despliegue de la aplicación con el objetivo de realizar pruebas a los estudiantes para constatar que los resultados no están limitados a la obtención del SEAI como software sino como herramienta de aprendizaje.
- Utilizar la aplicación como herramienta de solución a problemáticas similares a la descrita en este trabajo, de modo que se haga extensible su uso en otras asignaturas.
- Seguir el curso de la investigación con el fin de agregar nuevos módulos como el evaluador, planteado por algunos autores con el fin de aumentar el campo de aplicación del sistema.