Universidad de las Ciencias Informáticas

Facultad 2

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

Título: “Sistema para identificar y clasificar Estudiantes Potencialmente Talentosos en Informática.”

Autor:

Amado Naranjo Comas.

Tutores:

M.Sc. Manuel Villanueva Betancourt

Ing. Arianna Páez Valdés

Co-tutor:

Ing. Ernesto Avilés Vásquez

Ciudad de La Habana, Cuba, 2015

“Año 57 de la Revolución”

"Para nosotros esta es una escuela para formar talentos, no solo para recogerlos, sino para desarrollarlos, para prepararlos. Calidad humana y revolucionaria por encima del talento"

Fidel Castro Ruz (2002)

Declaración de Auditoría

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

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

_____

Amado Naranjo Comas

A nuestro líder histórico y artífice principal de este gran sueño que hoy disfrutamos Comandante Fidel Castro Ruz. Gracias a la Revolución por darnos esta oportunidad de crecer profesionalmente con el único interés de que continuemos forjando este camino de Revolución lleno de lauros y glorias. A la inspiración que siempre he tenido como figura única e irremplazable el Ché. Al MININT por toda la ayuda brindada en estos cinco años. A mis padres Belkiss y Amado por su ejemplo y enseñanzas tan valiosas que me han legado, a mis abuelos Ana, Miriam y Manuel por sus legendarios consejos y confianza en mí, a mis otros padres Giselle y Seguí por tratarme como un hijo más y compartir conmigo momentos únicos, a mis hermanos Xielem, Nadia y Osniel por ser siempre mis confidentes y darme ese cariño familiar que es siempre tan necesario, a mis tíos Olga y Manuel por extenderme la mano en momentos difíciles de mi vida y brindarme su apoyo, a mi familia de Santa Clara y muy especialmente a mi esposa Arianna por ser mi mayor crítica y hacerme sentir siempre el hombre más afortunado del universo.

A los profesores que de alguna forma u otra contribuyeron a mi formación como profesional, incluyendo a los miembros del tribunal, mi tutor Manuel Villanueva y cotutor Ernesto Avilés, a las valiosas amistades que he ganado en este pequeño pero arduo recorrido.

A todos muchas gracias.

A mi familia y muy especialmente a mis padres y esposa por todos mis resultados que también son de ellos. Nunca podré agradecerles lo suficiente por su apoyo y confianza.

Resumen

La identificación de Estudiantes Potencialmente Talentosos (EPT) es un proceso extremadamente complejo y cuidadoso que requiere principalmente la participación de expertos en materia de Pedagogía, pues en ello inciden aspectos subjetivos que pueden distorsionar la realidad y calidad de la selección. Actualmente se desarrolla en la Universidad de las Ciencias Informáticas, un proyecto de innovación pedagógica con el acrónimo de Talenmáticos, cuyo objetivo esencial es llevar una atención sistémica y científicamente fundamentada a aquellos estudiantes seleccionados como potencialmente talentosos. La manualidad en los procesos de identificación y clasificación aumenta el tiempo en que son identificados y crea divergencia entre los criterios de los expertos reduciendo así la posibilidad de identificar la mayor cantidad de talentos. El objetivo de este trabajo es proponer una herramienta informática utilizando técnicas de Razonamiento Basado en Casos y Algoritmos Conceptuales que permita en un corto tiempo identificar la mayor cantidad de EPT y ofrecer agrupaciones conceptuales basadas en las deficiencias de cada uno o grupo de ellos.

Palabras clave: algoritmos conceptuales, clasificación, estudiantes potencialmente talentosos, identificación, razonamiento basado en casos

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

La educación superior tiene entre sus objetivos específicos forjar profesionales que garanticen el desarrollo económico, técnico y social del país a partir de las necesidades de la fuerza calificada, que exige el mismo. Los Estudiantes Potencialmente Talentosos (EPT) detectados en la formación pregraduada, constituyen una fuente promisoria que a través de atención diferenciada, pueden alcanzar y liderar la búsqueda de nuevas soluciones científicas, pertinentes y sostenibles para afrontar los desafíos del creciente desarrollo. El proceso pedagógico debe estimular al máximo el desarrollo de todas las potencialidades que tienen los estudiantes universitarios a través de sus estudios de pregrado, tanto los que manifiestan un desempeño normal, como aquellos que evidencian un aprendizaje elevado y que pueden ser considerados como talentosos (Ortiz Torres, y otros, 2012).

Sobre la definición de Talento existen gran variedad de criterios con matices divergentes y que pueden generar diferentes interpretaciones por parte de investigadores y profesionales de la educación. Algunos de estos criterios (Castellanos y otros, 2003) exponen que el talento se expresa multifacéticamente a través de las diferentes actividades que realizan las personas en diferentes contextos sociales, donde existe una gran relación entre creatividad e inteligencia, también (Lorenzo García, 2010) se plantea que el talento es fruto de la interacción exitosa de tres componentes o dimensiones: la inteligencia, la creatividad y el compromiso con la tarea; en (Vergara, s/a) el término talentoso se reserva para definir a: "*aquel que alcanza una alta habilidad en un área específica…*" ; en (Petrovski, 1986) se considera que: "*el Talento es la combinación de capacidades que permiten llevar a cabo exitosamente alguna actividad laboral complicada, de manera propia y original*", entre otros.

Ante tanta diversidad conceptual se asume en el marco de este trabajo el criterio abordado en (Menéndez Pérez, y otros, 2012) donde se considera al Talento como "*La formación psicológica compleja en la cual las aptitudes y las actitudes interactúan dialécticamente entre sí y con el medio social, en el logro de resultados relevantes socialmente válidos.*"

En la Facultad 5 de la Universidad de las Ciencias Informáticas (UCI), se desarrolla un proyecto de innovación pedagógica con el acrónimo de Talenmáticos. El objetivo de este proyecto es formalizar y sistematizar un modelo de intervención educativa que permita brindar una atención sistémica y científicamente fundamentada desde el punto de vista psicopedagógico, a los EPT para su identificación, selección, estimulación y orientación (Menéndez Pérez, y otros, 2012).

La identificación de EPT constituye un proceso laborioso y extremadamente complejo incluso para expertos. En ello inciden aspectos subjetivos que en algunas ocasiones pueden distorsionar la realidad y calidad de la selección. La automatización de los procesos relacionados con la identificación y clasificación de EPT en Informática (EPT+I) en el Proyecto Talenmáticos puede garantizar una mayor *calidad*[1] de los resultados, con lo que se puedan tomar decisiones e incidir activamente en los Procesos de Enseñanza Aprendizaje (PEA) de los estudiantes seleccionados.

La confección de programas para los PEA de estos estudiantes, deben lograr la motivación, creatividad y el avance intelectual adecuado que impulse desde el perfeccionamiento hacia la competencia misma en la rama que se desempeñan. Uno de los principales problemas o barreras que interfieren para lograr el éxito de estos programas, y dar cumplimiento de sus objetivos, radica en la búsqueda de particularidades y características subyacentes esenciales que describen al EPT y su comportamiento. La obtención de rasgos que representen cualidades únicas del EPT o de un grupo de ellos, pueden brindar información cuantiosa acerca de sus gustos y tendencias, estilos de aprendizaje, características de su personalidad, aptitudes y capacidades, formas de motivación, que favorezcan la elaboración, desarrollo y aplicación de un PEA adaptado específicamente a las condiciones intelectuales y el aprovechamiento de las preeminencias intrínsecas de cada uno de ellos.

Entre las técnicas más específicas utilizadas para este tipo de trabajos, se encuentran las que permiten predecir, identificar y clasificar un resultado determinado simulando el razonamiento humano. A este rama de la Ciencia dedicada al desarrollo o uso de ordenadores con los que se intenta reproducir los procesos de la inteligencia humana se le denomina Inteligencia Artificial (IA) (Gálvez, 1998).

Actualmente el Proyecto Talenmáticos no cuenta con una herramienta informática que permita, ayude o agilice los procesos de identificación y clasificación de EPT+I en la UCI, como paso previo e indispensable, para una atención educativa y su posterior estimulación, orientación y gestión, incidiendo negativamente en la calidad de dichos procesos.

A partir de los elementos antes señalados, se enuncia como **problema a resolver**: ¿Cómo tributar al sistema de identificación y clasificación de los EPT que permita la atención educativa a esos discentes?

[1] Entendemos por calidad de identificación y clasificación de EPT: la disminución del tiempo requerido, la consideración de la mayor cantidad de criterios basados en la experiencia de expertos para obtener un resultado más exacto y la potencialidad para identificar y clasificar la mayor cantidad de EPT en Informática.

Del problema expuesto se determina como **objeto de estudio**: Técnicas de IA para la identificación y clasificación de objetos en un espacio de representación.

Se define como **objetivo general**: Desarrollar un sistema informático inteligente para identificar y clasificar los EPT de la UCI.

Hipótesis: con la aplicación del sistema informático inteligente desarrollado para la identificación y clasificación de EPT en la UCI, se contribuye a lograr un sistema de identificación en el Proyecto Talenmáticos de mayor rigor científico.

Como **objetivos específicos**:

1. Seleccionar técnicas de IA a utilizar, a partir de su caracterización en problemas de identificación y clasificación de un conjunto de objetos, así como las herramientas y lenguajes para el desarrollo del sistema informático.
2. Implementar un sistema informático inteligente que disminuya el esfuerzo humano necesario para identificar y clasificar los EPT en la UCI.
3. Realizar pruebas al sistema desarrollado.

En el **campo de acción:** la identificación y clasificación de EPT a través de técnicas de razonamiento basado en casos y algoritmos de agrupamiento conceptual respectivamente.

Para dar cumplimiento a los objetivos específicos de esta investigación se definen las siguientes **tareas de investigación**:

1. Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema actualmente.
2. Definición de los requisitos del sistema a desarrollar.
3. Representación del conocimiento (expertos) en una Base de Conocimientos (BC).
4. Desarrollo de las funcionalidades propuestas.
5. Prueba de las funcionalidades desarrolladas.

De los métodos utilizados en la investigación

Métodos Teóricos:

1. **Analítico Sintético**: en el análisis de documentos, trabajos de diplomas, sitios web donde se exponga información referente técnicas utilizadas en procesos de identificación y clasificación.

Se recopilaron los documentos de mayor relevancia para el desarrollo del presente trabajo de investigación.

2. **Sistémico-Estructural:** en el desarrollo del proceso de captura de requisitos del sistema, determinación de los actores, funciones y flujos de información
3. **Modelación**: en la representación de las categorías asociadas al desempeño intelectual de los discentes.

Métodos Empíricos:

1. **Consulta de la información en todo tipo de fuentes:** permite la elaboración del marco teórico de la investigación y sirve de guía para el resto del trabajo.
2. **Observación**: Con el desarrollo del **S**istema **I**nteligente **D**e **I**dentificación y **C**lasificación de Estudiantes Potencialmente **T**alentosos en Informática (SIDICT) se contrastan los resultados obtenidos (de la identificación y clasificación) con los resultados esperados (datos de expertos).
3. **Encuestas**: principalmente a profesores y miembros del Proyecto Talenmáticos con el objetivo diseñar una batería de encuestas que de la información de entrada al SIDICT.

Se plantea como **resultados esperados:**

1. Análisis de los fundamentos de la identificación y clasificación del talento en la educación superior, así como de los indicadores del talento informático.
2. Análisis del estado del arte de las principales técnicas en procesos de identificación y clasificación de un conjunto de objetos en un espacio de representación.
3. Sistema informático inteligente para la identificación y clasificación de EPT en la UCI.

Estructura Capitular:

Capítulo 1: Fundamentación teórica: En este capítulo se incluye un estudio del estado del arte de las diferentes estrategias utilizadas para la identificación y clasificación. Se realiza una evaluación crítica de las ventajas y desventajas de diferentes enfoques, destacando cuando deben usarse cada uno y se selecciona la técnica más adecuada, así como las herramientas y metodologías utilizadas para la implementación del sistema.

Capítulo 2: Diseño: Se realiza la propuesta del sistema a desarrollar, describiendo los elementos relacionados con el diseño de la solución planteada. Se crean los siguientes artefactos: diagrama de modelo de dominio, el diagrama de caso de uso del sistema, así como las descripciones, diagramas de clases y flujo de los principales casos de uso del sistema. Se define la arquitectura y los patrones de diseño a utilizar, conformándose finalmente el modelo de diseño.

Capítulo 3: Implementación y Prueba: Se implementan los elementos descritos en el capítulo anterior para alcanzar el objetivo general. Se llevan a cabo las pruebas para validar el sistema.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se realiza una caracterización de los distintos enfoques de la IA para problemas de identificación y clasificación de un conjunto de objetos en un espacio de representación. Se discuten sus principales ventajas y desventajas y se demuestran las potencialidades del Razonamiento Basado en Casos (RBC) y los Algoritmos de agrupamiento Conceptual (AC) como solución de la investigación. Se describen las características fundamentales de las herramientas y tecnologías de software utilizadas para el desarrollo del sistema.

1.2 Inteligencia Artificial

La IA se desarrolla actualmente como una de las ramas de las Ciencias de la Computación. En 1956 McCarthy propuso el término IA con el objetivo de agrupar todos los métodos y técnicas que se desarrollaban para simular el intelecto humano a través de equipos de cómputo. Esta técnicas aborda problemas poco estructurados, donde no se conoce de antemano cual es el mejor método para resolverlo (Rafael Bello, 2002).

No existe una definición de "Inteligencia Artificial" universalmente aceptada, pero con la intención de tener una aproximación conceptual para este trabajo se asume el siguiente concepto *"…la IA como la creación de programas de computadoras que emulen la forma de actuar y de pensar del ser humano, así como actuando y pensando racionalmente*" (Russel, 1995)

En el desarrollo actual de la IA existen técnicas para tratar problemas complejos mediante el empleo de procesos de búsqueda, formas de representación del conocimiento y aprendizaje. Estos métodos se dividen en dos grupos: la Inteligencia Artificial Simbólica (IAS) y la Inteligencia Artificial Conexionista (IAC).

El primero grupo de este campo comprende el desarrollo de diversas formas de representación del conocimiento en forma simbólica y explícita. Los métodos de solución de problemas usan este conocimiento basado en técnicas de búsquedas, los Sistemas Basados en Conocimiento (SBC), la sumarización lingüística, entre otros. El segundo grupo de este campo se identifica con las Redes Neuronales Artificiales (RNA). La dualidad entre estos criterios radica en el método específico para la

solución de un problema, que en la IAS la búsqueda se basa en atravesar un espacio de estado, mientras en la IAC es un proceso paralelo de cálculo de los niveles de activación de las neuronas.

1.3 Técnicas de IA para problemas de identificación y clasificación

1.3.1 Enfoque Lógico-Combinatorio para el Reconocimiento de Patrones

Por Reconocimiento de Patrones (RP), se podría identificar "*La zona del conocimiento que se ocupa del desarrollo de teorías, métodos, técnicas, y dispositivos computacionales para la realización de procesos ingenieriles, computacionales y/o matemáticos, relacionados con objetos físicos y/o abstractos, que tienen el propósito de extraer información que le permita establecer propiedades y/o vínculos de o entre conjuntos de dichos objetos sobre la base de los cuales se realiza una tarea de identificación o clasificación*." (Ruiz Shulcloper, 2013)

Este enfoque parte de la descripción de los objetos en forma de vectores y cada componente del vector representa uno de los rasgos analizados. La idea de la clasificación de objetos según este enfoque es la evaluación de los objetos haciendo uso de funciones de comparación de rasgos y funciones de semejanza entre las descripciones de los objetos. Algunos de los algoritmos más significativos en esta rama se encuentran:

- **Vecino más cercano**: Asigna a un objeto, la clase a la que pertenece el objeto que se encuentra más cercano a él.
- **Regla del máximo peso**: El nuevo objeto pertenece a la clase respecto a la cual tenga mayor peso informacional.
- **Regla del ideal**: El peso informacional de un objeto es maximal con respecto a su clase.

Este enfoque es muy popular en ramas de las ciencias poco formalizadas como la Medicina, la Ciencias Biológicas, las Geociencias, las Ciencias Sociales, entre otras. En estas disciplinas se presentan problemas de clasificación, de diagnóstico, de pronóstico, de determinación de factores, asociados a elementos intrínsecos de esas ramas como la información incompleta, grado de influencia de un fenómeno, soluciones múltiples, descripción de objetos mediante variables cuantitativas y cualitativas.

La selección de variables (Teoría de Testores) es también uno de los principales problemas en los que se investiga asociado a la importancia que se concede a identificar rasgos que mejor puedan describir un problema (Ruiz Shulcloper y otros, 1995). La Teoría de Testores como aplicación matemática al problema de selección de variables es una de las raíces más investigadas dentro del enfoque lógico-combinatorio.

Existen dos estrategias para el cálculo de todos los Testores Típicos (TT):

- **Algoritmos de escala interior**: Toman en cuenta la estructura interna de la matriz. Recorre un árbol binario, que es equivalente a recorrer el conjunto potencia de los rasgos (el conjunto de todos sus subconjuntos) o encontrar las condiciones que nos garanticen que determinadas columnas conforman un τ, en particular un TT.

- **Algoritmos de escala exterior**: A todo subconjunto $\Omega \subseteq P = \{X_1, ..., X_r\}$ le corresponde un n-uplo booleano característico $\omega = (\omega_1, ..., \omega_r)$ tal que el mismo toma valores iguales a 1 sólo en aquellas coordenadas correspondientes con elementos de R que estén en Ω. A estos n-uplos de ceros y unos se les hacen corresponder números binarios que a su vez representan números naturales. Cualquier orden que se defina en el conjunto de los números naturales N, se puede inducir en el conjunto potencia de R (el conjunto de todos sus subconjuntos), P(R).

La ventaja fundamental de utilizar TT en problemas donde los objetos están descritos por rasgos (cualitativos o cuantitativos), es poder discriminar aquellos que no brindan información suficiente y generen incertidumbres en la clasificación o interpretación de los resultados. (Naranjo Comas, 2014)

Muy apegada a esta Teoría de Testores, se encuentran los AC, cuyo objetivo es estructurar un conjunto de datos no solo por sus semejanzas cuantitativas sino también conceptuales. Los AC surgen como alternativa de la clasificación no supervisada, parte integrante del enfoque-lógico combinatorio para el RP. Su enfoque se compone de dos tareas fundamentales: el agrupamiento de entidades (*estructuración extensional*) y la caracterización, en la cual se determina el concepto de cada agrupación (*estructuración intencional*) (Pérez Suárez y otros, 2014).

Existen a su vez dos grandes grupos dentro de los algoritmos conceptuales:

- **Algoritmo Incrementales**: Basan su funcionamiento en la adaptación de los agrupamientos (o conceptos) con los nuevos objetos que se le van presentando, es decir, cada vez que se analiza un nuevo objeto este se clasifica, mediante una cierta estrategia, en uno de los agrupamientos ya existentes o se crean nuevos grupos. Por lo general, estos algoritmos se han desarrollado para los casos en que el conjunto de objetos a estructurar no está completamente dado, es decir, es dinámico. Ejemplos de estos algoritmos son: UNIMEM (Lebowitz, 1987), COBWEB (Fisher, 1987) y Galois (Carpineto & Romano, 1996).
- **Algoritmos No Incrementales**: Estructuran una muestra de objetos sin presuponer que estos se estructuran de uno en uno. Ejemplo: CLUSTER/PAF (Michalski, 1979), CLUSTER/2 (Michalski, 1983), CLUSTER/S (Stepp & Michalski, 1986), WITT (Hanson & Bauer, 1989) y K-Means conceptual (Ralambondrainy, 1995), LCconceptual (Martínez-Trinidad & otros, 2001) y RGC (Pons-Porrata & otros, 2002), entre otros.

En el enfoque lógico-combinatorio para el RP el aprendizaje se sustenta en la posibilidad de poder encontrar métricas y distancias adecuadas en dependencia de la naturaleza de los datos, aspecto que con frecuencia se vuelve difícil y depende de la subjetividad de los expertos y las características propias del problema. La alta dependencia asociada al conjunto de entrenamiento afecta la potencia del modelo en entornos donde se ven presentes el ruido y la imprecisión de los datos.

1.3.2 Árboles de decisión

Un árbol es un grafo dirigido sin ciclos, donde cada nodo tiene un nodo padre, excepto un nodo especial llamado raíz que no tiene padre. Los árboles se utilizan en su mayoría para representar jerarquías. Los árboles de decisión son clasificadores supervisados que se construyen a partir de un conjunto de objetos clasificados previamente. Para clasificar un nuevo objeto, se recorre el árbol según las propiedades que cumple el objeto, hasta llegar a un nodo hoja, donde se le asigna la clase. En el caso de los árboles de decisión no supervisados, no se tiene en cuenta a la clase a la que pertenece, porque los objetos no están etiquetados. El objetivo fundamental es que en el árbol quede representado correctamente el criterio de agrupamiento (Gutiérrez Rodríguez, 2012).

Entre los algoritmos más difundidos para esta técnica son:

- **Algoritmo CART**: Permite el trabajo con datos incompletos e incorpora estrategias de poda con el fin de obtener árboles donde se combine apropiadamente la capacidad de predicción con la cantidad de nodos. Utiliza métodos de “divide y vencerás”.
- **Algoritmo ID3**: Permite el tratamiento de datos simbólicos y sus extensiones C4.5 y C5.0 posibilitan el tratamiento de datos numéricos.
- **Algoritmos de Quinlan**: Basan su funcionamiento en una heurística hill-climbing y el criterio de selección de rasgos que usa se basa en el cálculo de la ganancia radial. Este cálculo puedo provocar indefiniciones o que rasgos con una baja ganancia sean seleccionados.

La ventaja más significativa de los árboles de decisión es que, pueden tratar atributos numéricos y nominales, son fáciles de usar, muchos de ellos son eficientes y existen variantes escalables a grandes volúmenes de datos.

Los algoritmos de aprendizaje de los árboles de decisión tienden a ajustarse demasiado a la evidencia, lo que puede traer como consecuencia que el modelo se comporte mal a la llegada de un nuevo objeto. Esta aproximación trae como consecuencia que el modelo pierda su carácter general y se vuelva muy específico, incidiendo negativamente sobre casos no visto. Además este tipo de técnicas son imprecisas con respecto a otros métodos como las redes neuronales y los sistemas basados en conocimiento.

1.3.3 Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) están inspiradas en las redes neuronales biológicas del cerebro humano. Estas están constituidas por elementos que se comportan de forma similar a la neurona biológica. Cada neurona es una unidad procesadora de información que recibe y combina señales desde y hacia otras neuronas. En el desarrollo de una red neuronal no hay que programar ni el conocimiento ni las reglas de procesamiento del conocimiento. La red neuronal aprende de las reglas del procesamiento del conocimiento mediante el ajuste de los pesos de las conexiones entre las neuronas de distintas capas de la red (Martín del Brío, 2001).

Las redes neuronales consisten en: un conjunto de entradas $X_j(t)$, pesos sinápticos de la neurona i,W_{ij} que representan la intensidad de interacción, reglas de propagación que proporcionan el valor del potencial postsináptico de la neurona *i* en función de sus pesos, una función de activación F_i que

proporciona el estado de activación actual y la función de salida que brinda la salida actual de la neurona *i* en función de su estado de activación.

Entre los modelos de red más difundidos se encuentran:

- **Adaline-Madaline (1960)**: Creada por Bernard Windrow para tareas de *predicción*. El sistema Madaline tiene una capa de unidades Adaline que están conectadas a una simple unidad Madaline. Las conexiones entre la capa de entrada y la capa de las unidades Adaline tienen asociadas un peso ajustable por cada una de ellas. Las conexiones entre la capa Adaline y la unidad Madaline no tiene asociado ningún peso. Cada unidad Madaline transmite su salida (-1 ó +1) a la unidad Madaline. El conjunto de entrenamiento para este sistema, es un conjunto de patrones de entrada emparejados con las salidas deseadas (supervisado). (Martín del Brío, 2001).

- **Back-Propagation (1985)**: Creada por David Parker para tareas de *clasificación*. La unidad procesadora en este tipo de red se caracteriza por realizar una suma ponderada de las entradas llamada S_j, presentar una salida a_j y tener un valor δ_j asociado que se utilizará en el procesos de ajuste de los pesos. El peso asociado a la conexión desde la unidad i a la unidad j se representa por w_{ij}, y es modificado durante el proceso de aprendizaje. Poseen un entrenamiento supervisado. El proceso de entrenamiento posee una fase de propagación hacia adelante y otra hacia detrás. El sistema NetTalk forma parte de uno de los casos de éxito en la aplicación de este tipo de red. NetTalk convierte texto escrito en inglés a voz de alta inteligibilidad. (Isasi Viñuela, 2003)

- **Hopfield (1982)**: Creada por John Hopfield para tareas de *optimización*. Posee una única capa de unidades procesadoras. Estas unidades se encuentran completamente interconectadas, cada unidad está conectada con todas las demás unidades, por lo que la convierte en una red recursiva. El aprendizaje en este modelo es no supervisado, pues no se conocen a partir de los vectores de entrenamiento la salida deseada.

- **Self-Organizing Maps (1979-1982)**: Creada por Kohonen para tareas de *conceptualización*. La red Self-Organizing Maps (SOM) tiene la característica de organizar mapas topológicos. El mapa que presenta la red a partir de la situación inicial aleatoria muestra las relaciones

existentes entre los diferentes patrones presentados a la red. La arquitectura de SOM la componen dos capas. La primera capa es la de entrada y la segunda la capa competitiva. Estas dos capas están totalmente interconectadas. Cada una de las conexiones tiene asociado un peso que será modificado a lo largo de la sesión de entrenamiento. Una de las diferencias de la red SOM con respecto a otros modelos de red, es que aprende sin supervisión, para luego trabajar en modo supervisado.

En general se puede resaltar las potencialidades de las RNA para gran variedad de problemas, especialmente los de clasificación. Sus desventajas están dadas por la imposibilidad de brindar información sobre las decisiones que toma una vez terminada la inferencia, además de no manejar idóneamente problemas cuyas variables están representadas en tipos Nominales, Ordinales y Discretos dificultando la configuración ideal de la red para la resolución de un problema concreto.

1.3.4 Sistemas de inferencia borrosa

Este tipo de inferencia es la lógica que utiliza expresiones que no son totalmente ciertas ni totalmente falsas. Los objetos pueden tomar un valor indeterminado de veracidad dentro de un conjunto de valores cuyos extremos son la verdad o falsedad absolutas.

La base del funcionamiento de los sistemas de inferencia borrosos son los conjuntos borrosos. Un conjunto borroso está caracterizado por una función de pertenencia, la cual asigna a cada elemento del dominio un grado de pertenencia a un conjunto en el intervalo [0,1].

Las *variables lingüísticas* constituyen el centro de la modelación de los sistemas de inferencia borrosa. Las variables lingüísticas y sus conjuntos borrosos se usan para describir relaciones entre variables en forma de reglas *Modus-Ponen* (if-then), conocidas también como reglas borrosas. Las reglas borrosas R se definen en forma de tupla (P, Q), donde P representa los conjuntos borrosos (*antecedentes*) y Q es el resultado (*consecuente*). La base de conocimiento en estos sistemas es una base de reglas borrosas (Piñeiro, 2005).

El método de evaluación de cada regla está basado en la aplicación de *TNormas* y el procedimiento para añadir nuevas reglas se basa en *CoNormas*. Entre los modelos de inferencia borrosa más difundidos se encuentran:

- **Mamdami**: El formato de esta regla es: *SI* u_1 es A_1 *Y* u_2 es A_2 *Y*…*Y* u_n es A_n *ENTONCES* b es B, donde los u_i y b son variables lingüísticas, y los A_i y B representan los valores lingüísticos (términos lingüísticos asociados a conjuntos borrosos) que dichas variables pueden asumir. Este método es más intuitivo, es ampliamente aceptado y se adapta mejor el lenguaje humano.
- **Takagi-Sugeno**: El formato seguido por esta regla es: *SI* u_1 es A_1 *Y* u_2 es A_2 *Y*…*Y* u_n es A_n *ENTONCES* b=f (u_1, u_2,…, u_n). Se mantiene la misma notación que en **Mamdami**, donde *f* representa la función lineal de las entradas. Este método es más eficiente en términos de computación, funciona bien con técnicas lineales, de optimización, adaptativas y de clasificación.

Sus deficiencias se encuentran en la base de reglas, si esta se construye solo a partir del criterio de los expertos puede no funcionar correctamente, debido a errores de localización de determinados puntos característicos en las funciones de pertenencia, respecto al número de reglas o respecto a determinadas áreas del espacio de búsqueda que se distinguen.

1.3.5 Sistemas Basados en Conocimiento

Un SBC es un programa de computadora que adquiere conocimiento especializado en un campo específico, para explotarlo mediante métodos de razonamiento que emulan el desempeño del experto humano en la solución de problemas (Peña Ayala, 2006).

Los SBC, están constituidos por una Máquina de Inferencia, una Base de Conocimiento, un Módulo de Explicación y un Módulo de Adaptación e Incorporación de Nuevo Conocimiento. La arquitectura de un SBC puede variar de acuerdo a las necesidades requeridas o deseadas. En la Figura 1.1 se muestra de forma general los componentes de la arquitectura de un SBC.

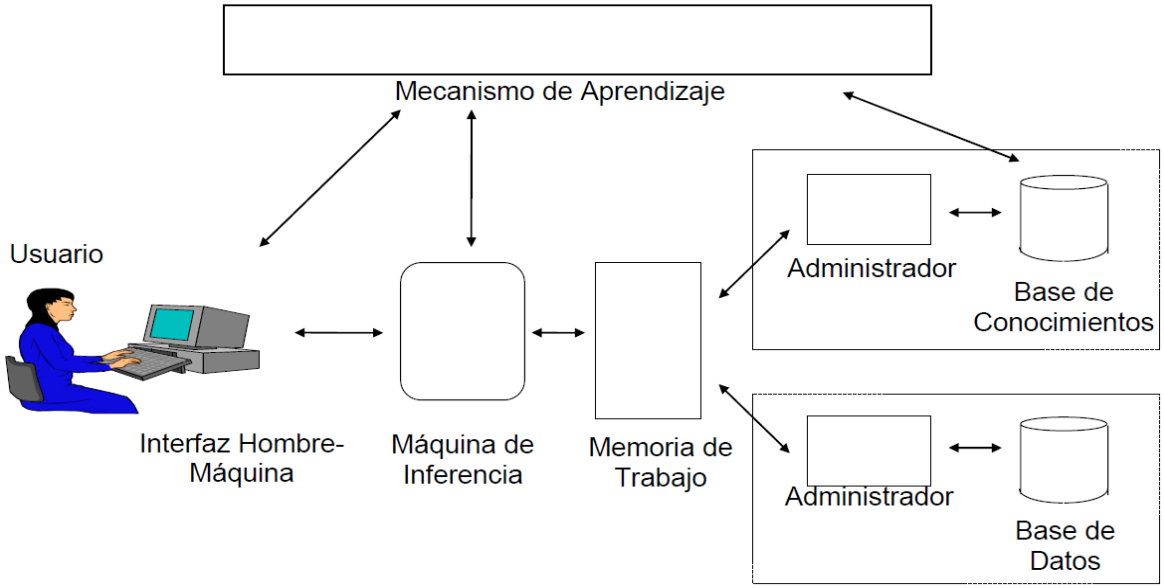

Figura 1.1 Arquitectura general de un SBC (Peña Ayala, 2006)

Existe una gran variedad de SBC, estos se diferencian por la forma de representar el conocimiento y por la naturaleza de la máquina de inferencia, algunos de ellos son: Sistemas Basado en Regla (SBR), Sistemas Basado en Probabilidad (SBP), Sistemas Basado en Frames (SBF) y el Razonamiento Basado en Casos (RBC).

Por la importancia que representa para este trabajo solo se analizarán los SBR y el RBC.

Los SBR se caracterizan por su forma de representar el conocimiento a través de reglas de producción y como método de inferencia utilizan *Modus-Ponen*. El proceso de razonamiento consiste en crear una cadena de inferencias que constituye un camino entre la definición del problema y su solución. Esta cadena de inferencias puede ser construida utilizando dos mecanismos diferentes: encadenamiento hacia adelante (forward) o encadenamiento hacia atrás (backward).

- **Forward**: Inicia el razonamiento a partir de un conjunto de hechos conocidos y progresa hacia la solución. Apropiada cuando existe bastante evidencia o la cantidad de conclusiones posibles es alta.

- **Backward**: Inicia el razonamiento a partir de una conclusión y prueba su validez a través de evidencias que la sustenten. Apropiada cuando existen pocas conclusiones o los datos de entrada no se adquieren automáticamente.

Los SBR tienen como ventaja fundamental su capacidad de interpretación y explicación del proceso de inferencia. Además cada regla es una unidad de conocimiento que puede ser añadida, modificada o removida independientemente de las otras reglas existentes (Modularidad), todo el conocimiento del sistema se expresa en el mismo formato (Uniformidad) y las reglas son un formato natural de expresar el conocimiento en algunos dominios (Naturalidad).

Se debe ser cuidadoso en la adición de un nuevo conocimiento a la base de reglas o en la modificación de este, pues puede resultar contradictorio y llevar a un encadenamiento infinito del proceso de inferencia. Otra dificultad está dada porque la ingeniería del conocimiento es difícil y la cantidad de reglas necesarias son numerosas cuando el dominio de aplicación es muy amplio, estos sistemas además no logran reconocer que reglas deben aplicar para cada ciclo de procesamiento, perdiendo así la perspectiva global al tener que analizar cada una de las reglas.

El RBC es una de las técnicas más usadas para construir SBC, en ellos un nuevo problema se resuelve a partir de *casos*[2] ya resueltos en el pasado. La arquitectura para estos sistemas consiste en una Base de Casos (BC), un procedimiento de recuperación de casos similares al nuevo, un procedimiento para adaptar las soluciones del caso recuperado a los requerimientos del nuevo problema y un mecanismo de aprendizaje, responsable de adquirir nuevos conocimientos y actualizar el existente.

El RBC basa su funcionamiento en una unidad mínima llamado caso, el cual tiene dos componentes: rasgos predictores (descripción del problema) y rasgos objetivos (solución del problema) (Peña Ayala, 2006).

Cada caso puede describir un suceso o una generalización se sucesos relacionados entre sí. En esta forma de solución de problemas se recupera un caso semejante el nuevo y la solución del problema recuperado se propone como solución potencial al nuevo problema.

[2] Descripción de un objeto o entidad, a través de atributos (numéricos, nominales y/o discretos)

Formalmente un caso C puede representarse como un conjunto de rasgos R = {$r_1, r_2, \ldots, r_n$}, donde cada r_i tiene asociado una función de comparación de rasgos δ_i, un conjunto de valores admisibles (dominio) y un peso W_i asociado a la relevancia de ese rasgo en el contexto. La comparación entre los casos se realiza a través de una función de semejanza β, y un *umbral de semejanza*[3] β_o.

Pasos para la inferencia basada en casos:

1. Presentar como entrada al sistema una descripción del problema actual (Presentación)
2. Recuperar el caso o casos pasados más similares al nuevo (Recuperación).
3. Reutilización de la información y conocimiento del caso o casos recuperados para resolver el nuevo problema (Adaptación).
4. Analizar la solución propuesta (Validación).
5. Almacenar la nueva solución una vez validada (Actualización).

Se pueden señalar como ventajas del RBC:

- Casos de éxito previos pueden ser utilizados para justificar nuevas soluciones.
- Casos sin éxito previos pueden ser utilizados para anticipar problemas.
- Si una situación se presenta repetidamente, la solución no se tiene que construir o generar desde cero.
- Se centra en las características más importantes del problema.
- Proponen soluciones en dominios no comprendidos totalmente por el sistema.
- El RBC es aplicable a una amplia gama de problemas.
- Es ideal cuando se torna difícil formular reglas.

Las desventajas del RBC son:

- No se explora todo el espacio de soluciones.

[3] Medida que especifica y garantiza que para pertenecer a una clase, el valor de semejanza (β) entre objetos tiene que ser significativo (mayor que el umbral de semejanza)

- Dependen de una adecuada función de semejanza.
- La consistencia entre los casos dificulta su mantención.

Algunos de los SBC más populares y difundidos sientan sus bases en estas dos técnicas (SBR, RBC) como pueden ser:

- **DENDRAL (1967)**: Se considera el primer SBC. Este sistema identificaba estructuras químicas moleculares a partir de su análisis espectro gráfico.
- **MYCIN (1970-1980)**: Se utilizaba para consultas y diagnóstico de infecciones de la sangre. El sistema introdujo la utilización de conocimiento impreciso para razonar y posibilidad de explicar el proceso de razonamiento**.**
- **PROSPECTOR**: Se utilizaba como explorador de yacimientos minerales partiendo de información descriptiva de una zona.
- **DELTA**: Sistema experto en reparación de locomotoras eléctricas y diesel.
- **IGC**: Utilizado para los diagnósticos de control de calidad.

1.4 Selección de las técnicas a utilizar

Luego de analizar el estado del arte relacionado a las diferentes estrategias para la identificación y clasificación de un conjunto de objetos, se toman el RBC en fusión con las técnicas de AC del enfoque lógico-combinatorio para el reconocimiento de patrones a ser utilizados en el desarrollo del SIDICT.

La selección se asienta en los fundamentos de la identificación del EPT en el Proyecto Talenmáticos, donde cada estudiante debe primeramente someterse a una encuesta (Instrumento[4]), cuyas preguntas responden a indicadores del talento. Utilizando RBC, los expertos pueden construir una BC utilizando casos de éxito (estudiantes ya identificados como EPT) y dejar a la herramienta la inferencia sobre los resultados de los estudiantes, en caso de algún fallo producido, se podrán ajustar los resultados a criterios de los expertos. Los algoritmos conceptuales jugarían un papel fundamental

[4] Un instrumento está compuesto por un conjunto de preguntas que atienden a diferentes dimensiones del talento (Cognitiva-Instrumental, Afectiva-Emocional, Psico-Social, Académico), representando una parte indispensable en el proceso de identificación de los EPT en la UCI, pues son los estudiantes los que dan respuestas a la preguntas de cada instrumento para luego ser analizados por los expertos del proyecto Talenmáticos.

en este proceso, su objetivo se centra en establecer dentro de los EPT identificados, agrupamientos conceptuales que brinden mayor información intra e inter grupal sobre las características subyacentes de cada estudiante, y así poder modificar y adaptar los Procesos de Enseñanza Aprendizaje (PEA) a estas peculiaridades.

En este caso los rasgos predictores serán los indicadores del talento y el rasgo objetivo la categoría de EPT (Sí/No).

1.5 Metodologías, lenguajes y herramientas de desarrollo

1.5.1 Metodología de desarrollo de software

Las metodologías de desarrollo de software abarcan todo el ciclo de vida del software, y se definen como “conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software.” (Patón, 2006-2007).

El Proceso Unificado de Desarrollo de Software (RUP) es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyecto.

El proceso unificado está basado en componentes, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectado a través de interfaces. El proceso unificado utiliza el lenguaje unificado de modelado (UML) para preparar todos los esquemas de un sistema software. (Jacobson, y otros, 2000)

Ventajas que ofrece trabajar con RUP (KRUP, 2000):

- Permite crear software que cumplan las expectativas de los usuarios a través de las especificaciones de requisitos.
- Permite llevar un seguimiento detallado en cada una de las fases de desarrollo.

Para el desarrollo del SIDICT, se decidió utilizar RUP, debido a su adaptabilidad[5] en el proyecto Talenmáticos, permitiendo generar la documentación necesaria para futuras versiones de la aplicación.

1.5.2 Lenguaje de Modelado

UML, (por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (Jacobson, y otros, 2000).

UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. Intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos, se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo (Jacobson, y otros, 2000).

Su utilización se debe a las siguientes ventajas:

- Posibilita la descripción de sistemas, simplificando la complejidad de estos y sin pérdida de información, haciendo posible la comprensión del sistema tanto para usuarios como desarrolladores.
- Brinda facilidades para el diseño, documentación, reutilización de código y detecciones de fallas.
- Facilita la comunicación entre desarrolladores, permite ahorrar tiempo en el desarrollo del software y hace más sencillas las modificaciones que se vayan a realizar.

[5] En este caso la adaptabilidad se refiere a aspectos intrínsecos del proyecto que hacen de RUP la metodología necesaria. Algunas de estos aspectos están dados por la construcción de varios sistemas que deben integrarse para conforman una plataforma más amplia incluyendo la estimulación, orientación y gestión de los EPT identificados, así como también para futuras investigaciones, le permitirá tener una visión más global de la herramienta a desarrolladores.

1.5.3 Herramienta CASE

1.5.3.1 Visual Paradigm

Visual Paradigm 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. 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 (Visual Paradigm, 2011).

Posibilita la representación gráfica de los diagramas permitiendo ver el sistema desde diferentes perspectivas, como el de componentes, despliegue, secuencia, casos de uso, clase, actividad, estado, entre otros. Además, identifica requisitos y comunica información, se centra en cómo los componentes del sistema interactúan entre ellos, sin entrar en detalles excesivos, además, permite ver las relaciones entre los componentes del diseño y mejora la comunicación entre los miembros del equipo usando un lenguaje gráfico.

Entre sus características más significativas se puede resaltar que su licencia es gratuita, posee varios idiomas y sus ediciones son compatibles.

1.5.4 Lenguajes de Programación

1.5.4.1 HTML

HTML (*Hyper Text Mark-up Language*) es un lenguaje de marcado principalmente usado para la creación de páginas web. Se usa para describir y traducir la estructura y la información en forma de texto, así como para ilustrar el texto con elementos como imágenes, videos, etc.

El código HTML se define mediante etiquetas (<et_inicio> </et_fin>), que permite añadir cualquier elemento a una página web. Permite también incluir scripts de otros lenguajes de programación como JavaScript o PHP, para así permitir la creación de páginas dinámicas.

La versión de HTML usada es la v.5.

1.5.4.2 CSS

CSS (*Cascading Style Sheets*) es un lenguaje formal que se utiliza para definir la presentación de un documento estructurado escrito en HTML o XML. El W3C (*World Wide Web Consortium*) es el organismo encargado de la especificación de las hojas de estilo que sirve como estándar.

La idea que se encuentra detrás de las hojas CSS es separar la estructura de un documento de su presentación. Cuando se usa CSS, las etiquetas de los documentos HTML o XML no proporcionan información de cómo ha de ser la presentación, sino que únicamente marcan la estructura del documento. La correspondiente hoja de estilo CSS se encarga de especificar como se ha de mostrar esa etiqueta: color, fuente, alineación del texto, etc.

Algunas de las ventajas de usar CSS son:

- Se tiene control centralizado de la presentación de la web que agiliza cualquier actuación.
- Los navegadores permiten a los usuarios usar su propia hoja de estilo, así aumenta la accesibilidad.
- Una página web puede disponer de diferentes hojas de estilo para diferentes dispositivos o para que el usuario pueda escoger.

Se utiliza para el diseño de las interfaces la versión CSS3.

1.5.4.3 JavaScript

JavaScript (JS) es un lenguaje de programación interpretado que se utiliza principalmente en páginas web. JS permite la orientación a los objetos. Todos los navegadores actuales interpretan el código JS integrado en las páginas web. El código JS se puede incluir en un documento HTML o en cualquier código que se acabe traduciendo a HTML en el navegador del cliente. El código JS es visible y puede ser leído por el usuario ya que se ejecuta en el navegador del cliente.

1.5.4.4 JQuery

Jquery es una biblioteca de JavaScript que permite simplificar la manera de interactuar con los elementos HTML. Existen multitud de librerías en JQuery que aplican efectos a diversos elementos de las páginas web. Entre las más usadas se encuentran *DataTables*[6], *HighCharts*[7], *JQuery-ui*[8], etc.

[6] Plugin de JQuery que permite hacer tablas dinámicas.

1.5.4.5 Python

Python es un lenguaje de programación interpretado creado por Guido van Rossum en 1991. Hoy en día Python se desarrolla como un proyecto de código libre administrado por la fundación se software Python (Python Software Foundation) (Python, 2015).

Python es un lenguaje multiparadigma, permite varios estilos como: programación orientada a objetos, programación estructurada y programación funcional. Este lenguaje se identifica por la identación, tipo dinámicos y gestión automática de memoria (Python, 2015).

Python se puede ejecutar sobre Windows, Linux/Unix y Mac OS, es libre de usar incluso para proyectos comerciales gracias a su licencia OSI (Iniciativa de Fuente Abierta) de código abierto. Viene acompañado de un intérprete interactivo que permite agilizar el desarrollo de programas, pues sirve de banco de pruebas de las nuevas ideas y así conocer rápidamente el resultado (Python, 2015).

Entre las ventajas más significativas y que distinguen a este lenguaje se encuentran:

- Sintaxis simple, clara y sencilla.
- Tipado dinámico.
- Gran cantidad de librerías disponibles.
- Lenguaje de alta potencia.

Python trabaja en entornos de desarrollos con bajo consumo de máquina lo que aumenta el rendimiento de las estaciones de trabajo. Actualmente gana aceptación en el mercado mundial de la industria del software por los beneficios que brinda, lo que lo convierte en un fuerte rival para otros lenguajes, asegurando de esta forma su continuidad de uso y actualización con los consiguientes aumentos en soportes y garantías.

La versión de Python utilizada para el desarrollo del presente trabajo es Python v2.7.

1.5.4.5.1 Framework Django

[7] Framework escrito en JQuery para mostrar en formas de gráficos dinámicos el contenido almacenado en la base de datos.
[8] Versión de JQuery con mejoras de diseño gráficas.

Django es un framework de desarrollo web de código abierto escrito en Python. Inicialmente fue desarrollado para gestionar varias páginas orientadas a noticias. Se publicó bajo la licencia BSD en Julio del 2005. En 2008 se creó Django Software Foundation que se hizo cargo de Django para darle proyección en el futuro (Django Project, 2015).

Las principales características de Django son:

- **Mapeador objeto-relacional**: las clases del modelo (base de datos) se definen en Python y se trabaja con la API de acceso a la base de datos que provee Django. Esto permite desarrollar independientemente el motor de la base de datos y evita en cierta medida el uso de SQL.
- **Vistas genéricas**: incorpora un sistema de vistas genéricas para hacer tareas habituales: listar registros, ver el detalle de un registro, borrar un registro, etc.
- **URLs elegantes**: permite crear URLs elegantes y limpias, admitiendo el uso de expresiones regulares.
- **Sistema de platillas**: incorpora un sistema de plantillas que permite separar el diseño gráfico y programación. Se puede editar el HTML sin tener que tocar el código Python.
- **Cache**: usa memcached para obtener buen rendimiento.
- **Aplicaciones pluggables**: las aplicaciones se pueden instalar en cualquier otro proyecto Django, es decir, son reutilizables.

La versión de Django utilizada para el desarrollo del presente trabajo es Django v1.6.

1.5.5 Sistema Gestor de Base de Datos PostgreSQL

PostgreSQL es un motor de base de datos creado en 1986 con el lanzamiento de su primera versión. Está distribuido bajo licencia BSD (Berkeley Software Distribution) y con su código fuente está disponible libremente. Cuenta con más de 15 años de desarrollo activo y una arquitectura probada, que se ha ganado una sólida reputación de fiabilidad e integridad de datos. Funciona en los principales sistemas operativos, incluyendo Linux, UNIX, Mac OS y Windows (Postgresql, 2013).

Algunas de sus características principales son:

- Uso de funciones disparadoras (trigger).

- Uso de funciones agregadas de ventana.
- Multiplataforma.
- Extensible.
- Alta concurrencia.
- Estabilidad y confiabilidad.
- Instalaciones ilimitadas.
- Diseñado para ambientes de grandes volúmenes de datos.

La utilización de este gestor de base de datos se hace potencial para el desarrollo del SIDICT dada sus ventajas de uso libre, el soporte para varias plataformas aumentado así su eficacia y aplicación en diferentes sistemas operativos.

1.5.6 Servidor Web

Un servidor se puede definir como un software que ejecuta acciones a petición de los usuarios brindándole servicios. El término también puede ser asociado a la máquina en la cual funciona este software y que brinda datos a máquinas clientes. Por tanto la referencia a un servidor web puede traer confusión, ya que se puede interpretar como la computadora que almacena y gestiona los sitios web, en este caso pueden ser utilizadas para ofrecer hosting u hospedaje. También se puede llamar servidor web al software que funciona en la máquina y maneja la entrega de los componentes de la página web a través del protocolo HTTP (López Pino, 2011)

1.5.6.1 Nginx

Servidor web de altos rendimientos que posee características que posibilitan crear una infraestructura web moderna y eficiente. Con su principal característica, como balanceador de carga, es capaz de detectar la caída de uno de sus servidores reales, dejando de enviarle peticiones a este y repartirlas entre los demás nodos del clúster, utiliza el mecanismo nodo director donde todos las peticiones pasan a través de él antes de ser enviadas a los servidores web, hace uso de revese proxy acelerado sin caché, es conocido por su estabilidad, gran conjunto de características, configuración simple, y bajo consumo de recursos. (López Pino, 2011)

1.5.7 Conclusiones parciales

En el presente capítulo se dio cumplimiento al primer objetivo específico sobre la caracterización de las técnicas de Inteligencia Artificial más usadas en problemas de identificación y clasificación de un conjunto de objeto en un espacio de representación y la selección de las más idóneas y con mayor valor potencial en esta investigación: el Razonamiento Basado en Casos (para la identificación de los EPT) y los Algoritmos de Agrupamiento Conceptual (clasificación no supervisada y descripción conceptual de cada grupo formado a partir de los EPT identificados), se seleccionaron además para cada una de las técnicas, modelos específicos que se adapten mejor a las particularidades del problema.

Se definió como metodología de desarrollo a utilizar RUP por las características particulares existentes en el proyecto. El lenguaje de modelado (UML 1.x), apoyándose en la herramienta CASE Visual Paradigm v8.0. Además, se decidió utilizar como lenguaje de programación Python v2.7, usando el framework de aplicaciones web Django v1.6. Se seleccionó también Nginx como servidor de aplicaciones web.

Capítulo 2: Diseño del sistema

2.1 Introducción

El presente capítulo hace referencia a la presentación de la propuesta de solución, con el objetivo de comprender y esclarecer a mayor detalle el problema de investigación. Se representa el modelo de dominio, se enumeran los requerimientos funcionales y no funcionales. Posteriormente se realiza la descripción de los actores y los principales casos de uso del sistema, incluyendo los diagramas de clases y secuencia correspondientes.

2.2 Propuesta del Sistema

El desarrollo del SIDICT será de gran utilidad en los procesos de toma de decisiones para los expertos del Proyecto Talenmáticos de la Universidad de las Ciencias Informáticas, por sus posibilidades de análisis masivo de los datos de estudiantes y sugerencias sobre los resultados obtenidos. Las funcionalidades principales del sistema se encuentran en la identificación y clasificación de EPT. Se le llama identificación de EPT en el marco del Proyecto Talenmáticos, al proceso sinérgico que debe existir entre expertos (miembros del proyecto, encargados de realizar la identificación a partir de sus experiencias y conocimientos de pedagogía y/o informática), profesores y estudiantes para lograr éxito en la identificación de EPT. Más detalladamente se puede decir que este proceso se compone de varios pasos como:

1. Se procede a elaborar un Instrumento (semejante a una encuesta), compuesto por preguntas que se enfocan en evaluar aspectos sobre las dimensiones de talento (Cognitiva-Instrumental, Psico-Social, Afectivo-Emocional y Académicos).
2. Los expertos luego de poseer un consenso sobre el Instrumento a aplicar, proceden a enviar este documento a los profesores (guías de grupo o líderes de proyecto).
3. Una vez recibido este Instrumento, los profesores deben enviarlo a cada uno de los estudiantes que estén bajo su responsabilidad, mando o custodia.
4. Los estudiantes responden las preguntas del Instrumento y la devuelven a sus respectivos profesores.
5. Los profesores antes de volver a enviar las respuestas de los Instrumentos a los expertos, deben revisar minuciosamente que cada estudiante ha sido honesto con sus respuestas.

6. Una vez que los profesores envían el resultado de cada Instrumento, los expertos comienzan con análisis de los mismos hasta lograr el resultado final (la identificación de todos los EPT).

Durante el desarrollo del SIDICT y por argumentos mencionados en el capítulo anterior se decidió utilizar para esta tarea el RBC. El SIDICT realiza de forma muy particular estos pasos, analizando los datos obtenidos en las encuestas de los estudiantes y los casos previamente introducidos en la base de conocimiento y realiza la inferencia clasificando a los estudiantes en cuatro categorías. En la primera (Categoría **A**) se muestran los estudiantes que se consideran EPT, en la segunda (Categoría **B**) los estudiantes que se encuentran muy cerca de ser considerados EPT, en la tercera (Categoría **C**) los estudiantes que no son EPT y la cuarta (Categoría **D**) muestra los estudiantes para los cuales no existen casos semejantes.

La clasificación de EPT ya identificados, consiste en establecer grupos de estudiantes que tengan dificultades semejantes en los resultados obtenidos del proceso anterior. Esto permitiría a los expertos tener un conocimiento o visión más global, sobre los puntos en que deben incidir a través de los PEA e influenciar mediante la estimulación y orientación de los mismos.

El SIDICT utiliza la clasificación no supervisada (agrupamiento), específicamente el agrupamiento conceptual, para llevar a cabo esta tarea. Tarea donde no se conocen cuantos grupos de EPT deben existir, ni se tienen definidos ejemplos que pertenezcan a un grupo determinado y es de gran necesidad agruparlos y brindar un conjunto de características que permitan a los expertos, conocer que preguntas asociados a cada dimensión del talento es donde mayor dificultad hay y el nivel de dificultad, además de los grupos de EPT donde esta dificultad es preponderante.

En el epígrafe 2.4.4, se describen a mayor detalle algunas de las particularidades de las técnicas utilizadas.

2.3 Modelo de Dominio

El modelo del dominio muestra los modeladores clases conceptuales más significativos en un dominio del problema; es el artefacto más importante que se crea durante el análisis orientado a objetos. *"Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés."* (Larman, 1999)

A continuación se muestra el diagrama de clases del modelo de dominio:

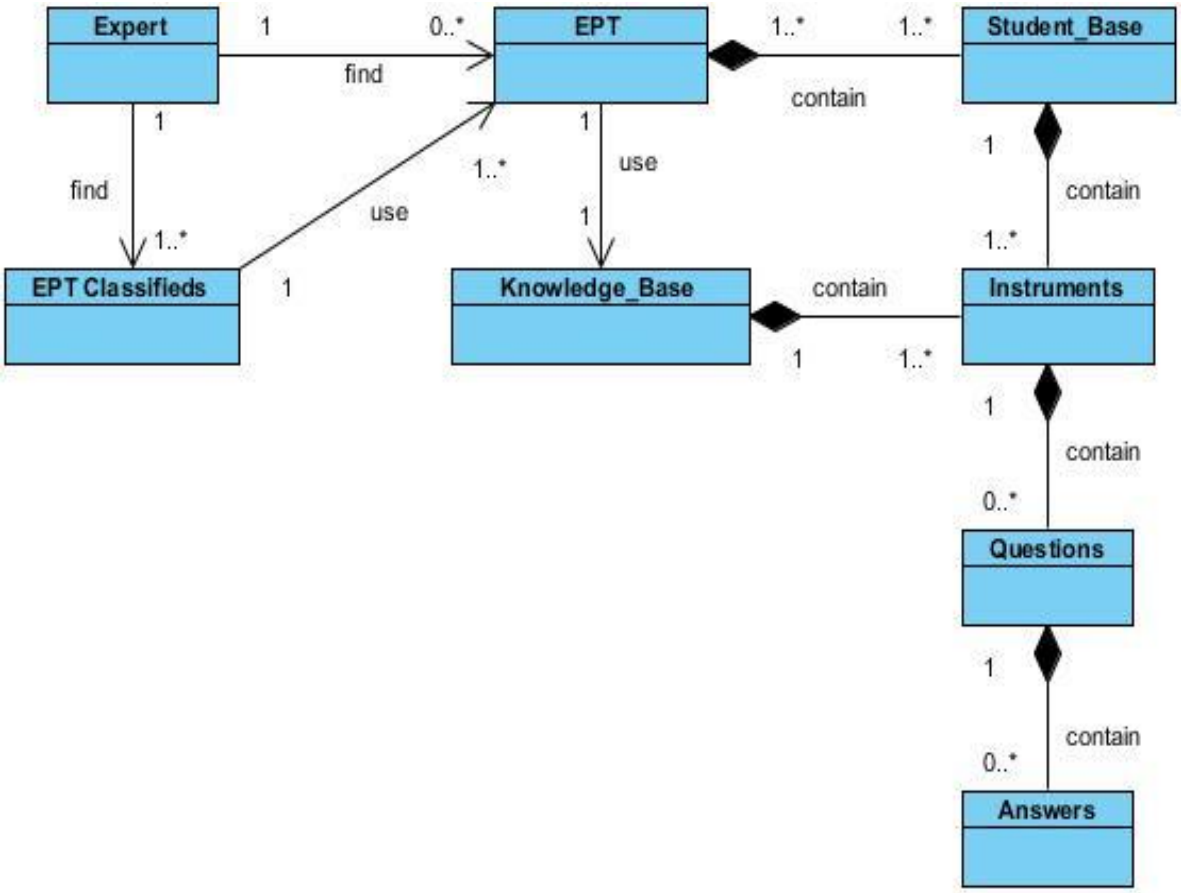

Figura 2. 1 Proceso de Identificación y Clasificación de EPT

Descripción de las clases del dominio:

Expert: Usuario con suficientes privilegios (o Administrador) que puede realizar la identificación y clasificación de los EPT.

EPT: Estudiantes identificados positivamente como EPT.

EPT Classifieds: Conjuntos de EPT agrupados por sus semejanzas.

Knowledge_Base: Resultados que pueden representarse como casos de éxito o casos de fallo.

Student_Base: Conjunto de estudiantes que serán analizados.

Instruments: Está formado por un conjunto de preguntas, que obedecen a las dimensiones del talento (Cognitiva-Instrumental, Afectiva-Emocional, Psico-Social, Académico).

Questions: Preguntas asociadas a cada dimensión del talento.

Answers: Respuesta a la pregunta.

2.4 Requisitos del Sistema

Un requisito funcional es una declaración de los servicios que debe proporcionar un sistema, de manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportarse en situaciones particulares. (Sommerville, 2005)

A continuación se enuncian los Requisitos Funcionales (RF) definidos:

2.4.1 Requerimientos funcionales

RF1. Autenticar usuario.

RF2. Modificar contraseña.

RF3. Solicitar nueva contraseña.

RF4. Ver mi perfil de usuario.

RF5. Actualizar mi perfil de usuario.

RF6. Listar usuarios.

RF7. Gestionar usuarios.

RF7.1. Insertar nuevo usuario.

RF7.2. Actualizar datos de usuario.

RF7.3. Eliminar usuario.

RF8. Listar base de conocimiento.

RF9. Gestionar base de conocimiento.

RF9.1. Insertar nuevo caso.

RF9.2. Ver detalles del caso.

RF9.3.Eliminar caso.

RF10. Listar Instrumentos.

RF11. Gestionar Instrumentos.

RF11.1.Insertar nuevo Instrumento.

RF11.2.Actualizar datos del Instrumento.

RF11.3.Eliminar instrumento.

RF12. Ver detalles de un Instrumento.

RF13. Listar estudiantes.

RF14. Gestionar estudiantes.

RF14.1.Insertar nuevo estudiantes.

RF14.2.Actualizar datos del estudiante.

RF14.3.Eliminar estudiante.

RF15. Ver detalles del estudiante.

RF16. Identificar EPT.

RF17. Clasificar EPT.

RF18. Ver detalles de identificación.

RF19. Seleccionar como EPT.

RF20. Añadir estudiante como conocimiento.

RF21. Listar EPT.

RF22. Ver detalles del EPT.

RF23. Listar EPT clasificados.

RF24. Ver detalles de grupos de EPT.

RF25. Contactar administrador.

2.4.2 Requerimientos no funcionales

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

1. **Usabilidad:**
   - El sistema también podrá ser usado por aquellas personas que no poseen conocimientos avanzados en informática.
   - Permite al acceso a cada funcionalidad de forma fácil e intuitiva.
   - Incluye un manual de ayuda que permite establecer los conocimientos básicos para la correcta utilización del sistema.
2. **Seguridad:**
   - Se presenta como primera acción del sistema la autenticación, cada usuario debe ingresar su nombre de usuario único y una clave. Además cada usuario tiene un rol asignado (Administrador o Usuario normal) y cada administrador puede asignar permisos a los usuarios para que puedan realizar determinadas acciones a las que no están autorizadas.
   - Confidencialidad: Se debe controlar el manejo de los permisos, para que los usuarios únicamente puedan acceder a la información a la que están autorizados.
   - Integridad: Se debe controlar el manejo de la información, para que solo aquellas personas con permiso puedan modificar la información, evitando así posibles alteraciones en los resultados de la identificación y clasificación.
   - Disponibilidad: Debe garantizarse el acceso de cada usuario del sistema las 24 horas del día.
3. **Confiabilidad:** El sistema debe guardar automáticamente la información existente en caso se algún fallo que provoque la caída del sistema.
4. **Interfaz de usuario**: La interfaz de usuario debe poseer un diseño sencillo con una interfaz amigable donde predominen colores claros.

5. Interfaz de software:

Es necesario que el servidor este corriendo con estas características:

- Windows Server 2000 (o superior), Linux/Unix (cualquier distribución) o Mac OS.
- Python v2.5 (o superior) y Django v1.6 (o superior).
- Servidor de base de datos PostgreSQL v9.2.4.1

En las computadoras clientes:

- Un navegador web: Firefox 3.0 (o superior), Opera 8 (o superior), Chrome 6 (o superior), Internet Explorer 9 (o superior), Safari 10 (o superior).

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

- Procesador Intel Pentium IV 2.8 GHz (o superior), o variante AMD Athlon 64x2 (o superior).
- Memoria RAM de 512 Mb (o superior)

2.4.3 Descripción de la Base de Conocimiento (RBC)

En este tipo de sistemas (RBC), la base de conocimiento es una colección de casos (base de casos) y el método de solución de problemas consiste esencialmente en la recuperación de los casos semejantes a la descripción del problema a resolver y la adaptación de las soluciones dadas a ellas para construir la solución del nuevo problema.

Sea $\boldsymbol{C} = \{C_1, C_2, C_3, \ldots, C_n\}$ una colección de casos descritos en términos de un conjunto de atributos o rasgos (preguntas asociadas a las Instrumentos) $\boldsymbol{R} = \{R_1, R_2, R_3, \ldots, R_n\}$. Se entenderá por representación del caso i-ésimo C_i el l-uplo $X_i = (R_1(O_i), \ldots, R_n(O_i))$, donde $R_n(O_i)$ es el valor que toma el rasgo (pregunta) R_n en el objeto O_i. Para la presente investigación los valores de los rasgos serán únicamente Nominales (Muy Bajo (1), Bajo (2), Medio (3), Alto (4), Muy Alto (5)).

2.4.4 Proceso de Razonamiento (Identificación y Clasificación)

Para el proceso de identificación de los EPT, se utiliza el modelo de RBC expuesto en (Cordero Morales, 2013). Se exponen a continuación las características principales del modelo:

$$\beta(O_i, O_j) = \frac{\sum_{i=1}^{n} w + \delta_r(X_r(O_i), X_r(O_i)) + [1 - (I(X_r(O_i)) - I(X_r(O_j))]}{\sum_{i=1}^{n} w} \quad 1$$

donde $\beta(O_i, O_j)$: define la semejanza entre los objetos O_i y O_j, w define el peso asociado a cada rasgo por su relevancia en el contexto del problema, $\delta_r\ (X_r\ (O_i),\ X_r(O_j))$ función de comparación entre los objetos O_i y O_j; $I\ (X_r\ (O_i),\ X_r\ (O_j))$; incertidumbre asociada al rasgo r en los objetos O_i y O_j respectivamente.

La incertidumbre de los rasgos, pueden ser especificadas en caso de que los expertos la conozcan, de forma contraria, se propone calcular la desviación estándar (σ) de cada rasgo como valor de incertidumbre y luego normalizarlo en el conjunto difuso [0; 1], donde 1 representa la incertidumbre total (desconocimiento total) del rasgo y 0 conocimiento pleno, a partir de la función: (Naranjo Comas, 2014)

$$I(X_r) = \sigma(X_r) / máx\{X_r\} \quad 2$$

donde $I\ (X_r)$: define la incertidumbre del rasgo r, $\sigma\ (X_r)$ desviación estándar del rasgo r y $máx\{X_r\}$ valor máximo para el rasgo X.

Se utiliza también la utilidad esperada del caso, este criterio se basa en la semejanza obtenida a través de la función (1) y la incertidumbre en (2).

$$\upsilon(O_u) = \alpha\beta(O_n, O_r) + (1 - \alpha) * \theta(O_r) \quad 3$$

donde $\mu\ (O_u)$: define la utilidad esperada del caso, $\beta\ (O_n,\ O_r)$ semejanza del nuevo caso O_n y el caso recuperado O_r, $\theta\ (O_r)$ valor de incertidumbre del caso recuperado, α es un parámetro que cuando tiende a 1 le da más importancia a la semejanza y si tiende a 0 a la incertidumbre.

Para el proceso de clasificación de los EPT previamente identificados, se utilizan diferentes aspectos del modelo de AC expuesto en (Naranjo Comas, 2014), donde lo más significativo se enmarca en la función de calidad sobre el concepto construido:

$$\psi_i(TT_i) = \frac{\sum_{i=1}^{|T_i|} w(X_i)}{\sum_{i=1}^{|T_i|} I(X_i)} \quad 4$$

donde $\Psi_i(T_i)$: calcula la suma pesada de los rasgos que componen a cada Testor (T_i) obtenidos a través de la Matriz Básica (MB), $w(X_i)$ pesos del rasgo *i* asociado al T_i, $I(X_i)$ incertidumbre del rasgo *i* que pertenecen al T_i. Luego se seleccionan los $|\ máx\ \{\Psi_i(T_i)\}\ | = 3$, para formar los conceptos.

En caso hipotético donde suceda que, la suma de la cardinalidad de los *T* sea menor a 5 ($|T_1| + |T_2| +\ldots+ |T_n| < 5$), se introduce un umbral de ruido (error) para seleccionar aquellos rasgos que junto a los *T*, ayuden a diferenciar de manera conceptual cada agrupación formada. Este umbral, describe que los errores inferiores al umbral, presentan poca confusión y por ende pueden ser utilizados para establecer conceptos en las clases. De ser necesario el cálculo de este valor, influye en la cantidad de representaciones conceptuales que tendrá asociado cada agrupamiento, ayudando así a una mayor comprensión, y se define como:

$$\beta_r = \frac{\sigma(e(X_r))}{|U|} \quad 5$$

donde β_r: define el umbral de ruido, $|U|$ cantidad de objetos representados en el universo, $e(X_r)$ magnitud de confusión asociada al rasgo *r*.

Además para evaluar la calidad de los conceptos construidos, se utiliza la función:

$$M(K_i) = \frac{\sum_{\tau=1}^{|T_i|} 1 - I(X_\tau)}{|T_i|} \quad 6$$

donde $M(K_i)$: define el grado de certeza asociado al concepto construido de clase K_i, $I(X\tau)$ incertidumbre asociado al rasgo $X\tau$, que pertenece al T_i, $|T_i|$ cardinalidad del T_i.

2.4.5 Descripción de los actores del sistema

Un actor es una entidad externa que interacciona con el sistema participando en un caso de uso. Los actores no representan a personas físicas o a sistemas, sino su rol. Esto significa que cuando una persona interactúa con el sistema de diferentes maneras (asumiendo diferentes papeles), estará representado por varios actores.

Actor	Descripción
User	Posee privilegios para editar su perfil, ir a los rankings de EPT, ver resultados de identificación y clasificación, buscar un EPT, cambiar contraseña, pedir una nueva contraseña.
Admin	Incluye los privilegios de usuario, añadiendo las operaciones para gestionar el EPT, gestionar los usuarios, crear nuevos usuarios, realizar la identificación y clasificación de EPT y gestionar la BC. Un experto puede ser Admin del sistema.

Tabla 2. 1 Descripción de los actores del sistema

2.4.6 Definición de los casos de uso del sistema

Un caso de uso es una descripción narrativa de un proceso de dominio. (Larman, 1999)

Una vez identificados los actores del sistema se pasa a la elaboración del diagrama de casos de uso del sistema, representado en la Figura 2.2.

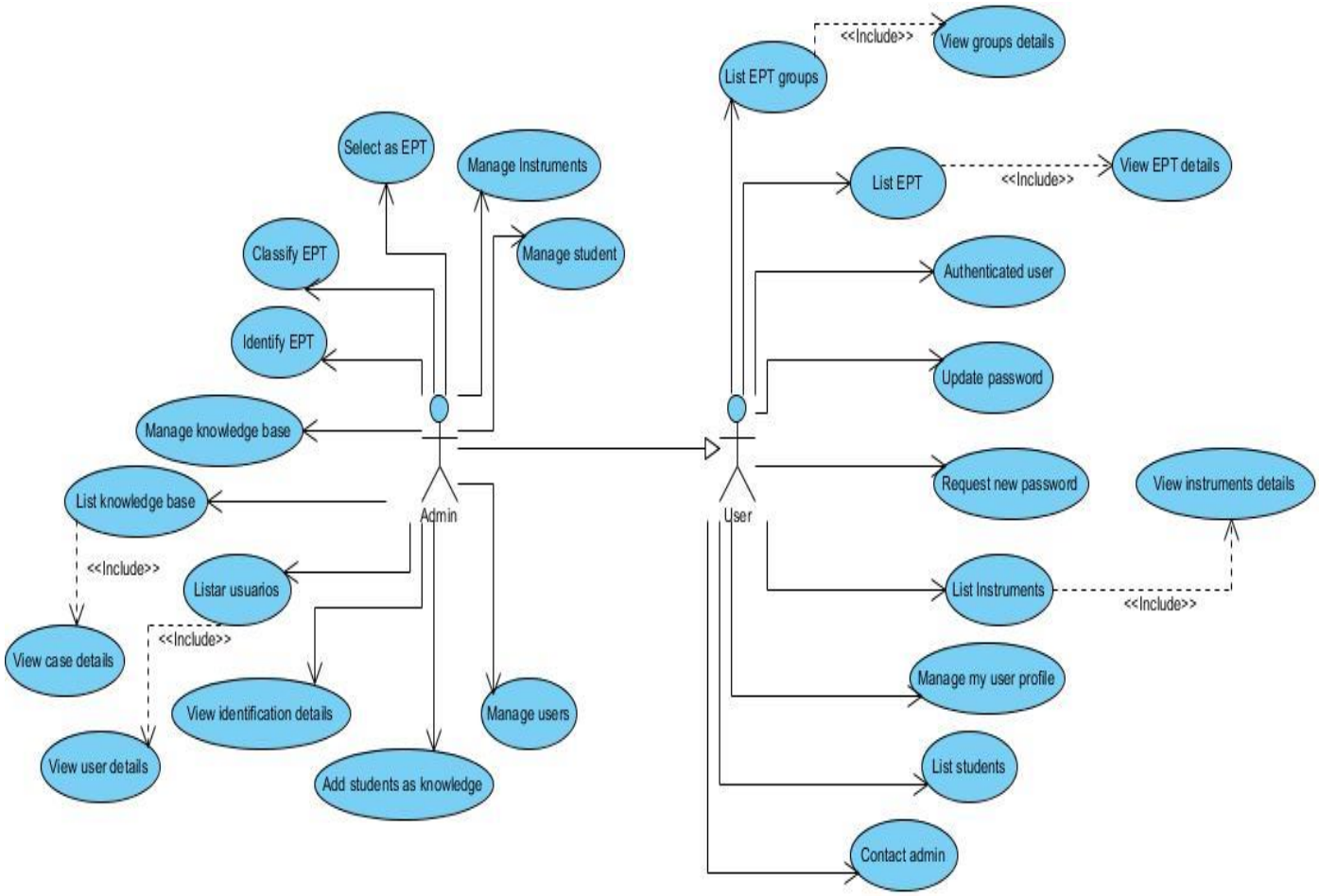

Figura 2. 2 Diagrama de Casos de Uso

2.4.6.1 Especificación de Casos de Uso

Se detallan a continuación las descripciones de los casos de uso más importantes concernientes al tema de investigación realizado.

CU1.Identificar EPT:

Nombre	Identify EPT
Actores	Admin
Resumen	El caso de uso inicia cuando el administrador selecciona que desea Identificar EPT y finaliza mostrando los detalles del proceso realizado.
Complejidad	Alta

Prioridad	Alta	
Referencias	RF16, RF18	
Precondiciones	Deben existir estudiantes en la base de datos.	
Postcondiciones	El sistema muestra los detalles del proceso realizado.	
Flujo de eventos		
Flujo básico <Identificar EPT>		
	Actor	**Sistema**
1.	Selecciona la opción Identificar EPT.	
2.		Muestra un mensaje: “Esta operación puede tardar varios minutos”. Desea continuar.
3.	Selecciona la opción “Aceptar” para continuar el proceso de identificación.	
4.		Obtiene los datos sobre los resultados de los instrumentos (encuestas) de los estudiantes y los datos de cada caso existente en la BC.
5.		Realiza la identificación de EPT a través de los datos recopilados y haciendo uso de funciones y criterios del RBC.
6.		Muestra un listado con los detalles del proceso culminado, mostrando al administrador los estudiantes que el sistema considera EPT (Categoría **A**), los que se encuentran cercanos a ser EPT (Categoría **B**), los que no se consideran EPT (Categoría **C**) y aquellos estudiantes para los cuales no existen casos semejantes en la base de casos (Categoría **D**).
7.		Marca aquellos que considera que son EPT y le indica la certeza de identificación.

8.		Termina el caso de uso.
Flujo alterno del flujo básico < Identificar EPT >Cancelar		
Actor		**Sistema**
3.a	Indica que no desea continuar con la Identificación de EPT y termina el caso de uso.	
Relaciones	**CU Incluidos**	CU Identificar EPT.
	CU Extendidos	No aplica.
Requisitos no Funcionales	No aplica.	
Asuntos pendientes	No aplica.	
Prototipo de Interfaz		

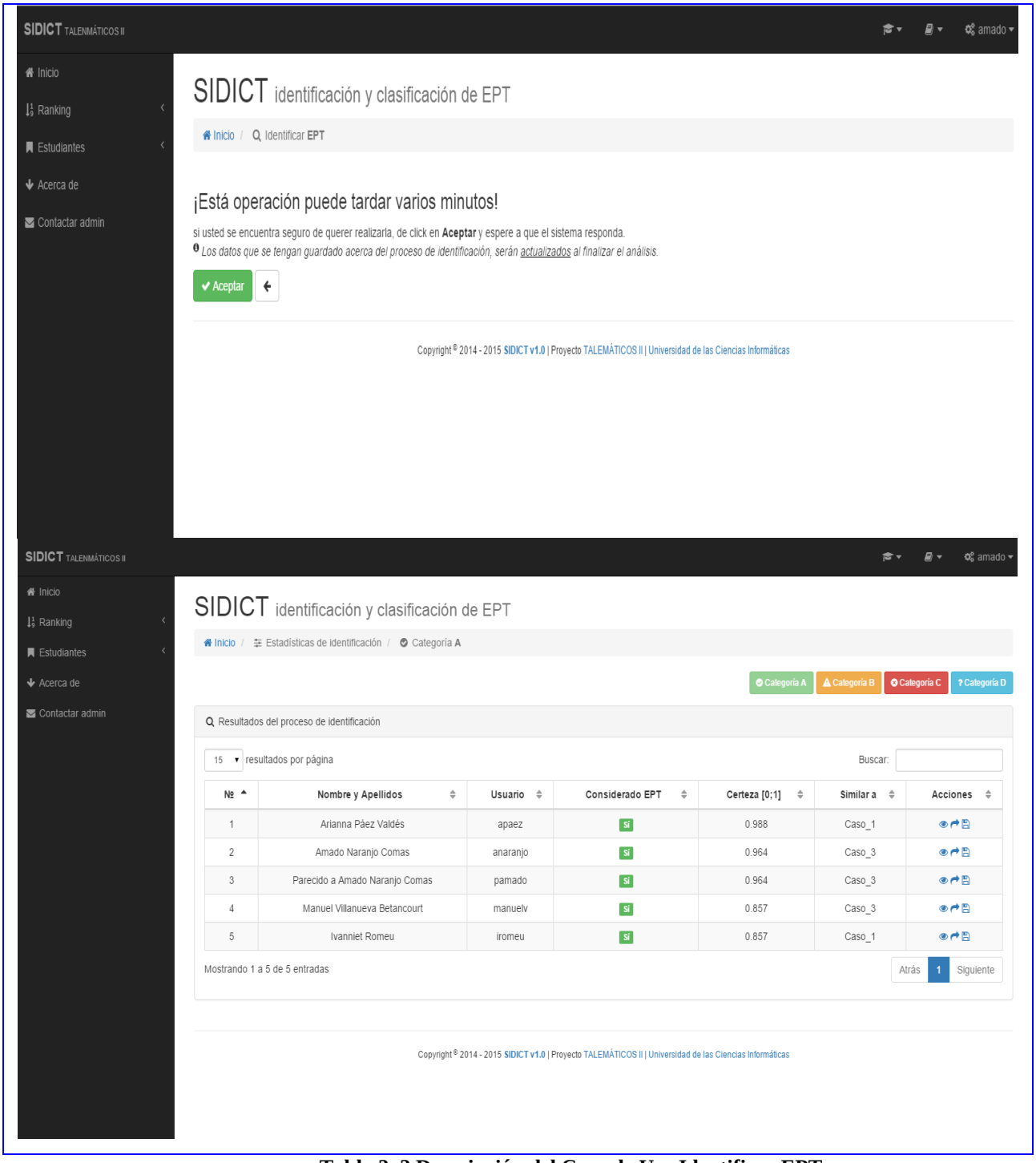

Tabla 2. 2 Descripción del Caso de Uso Identificar EPT

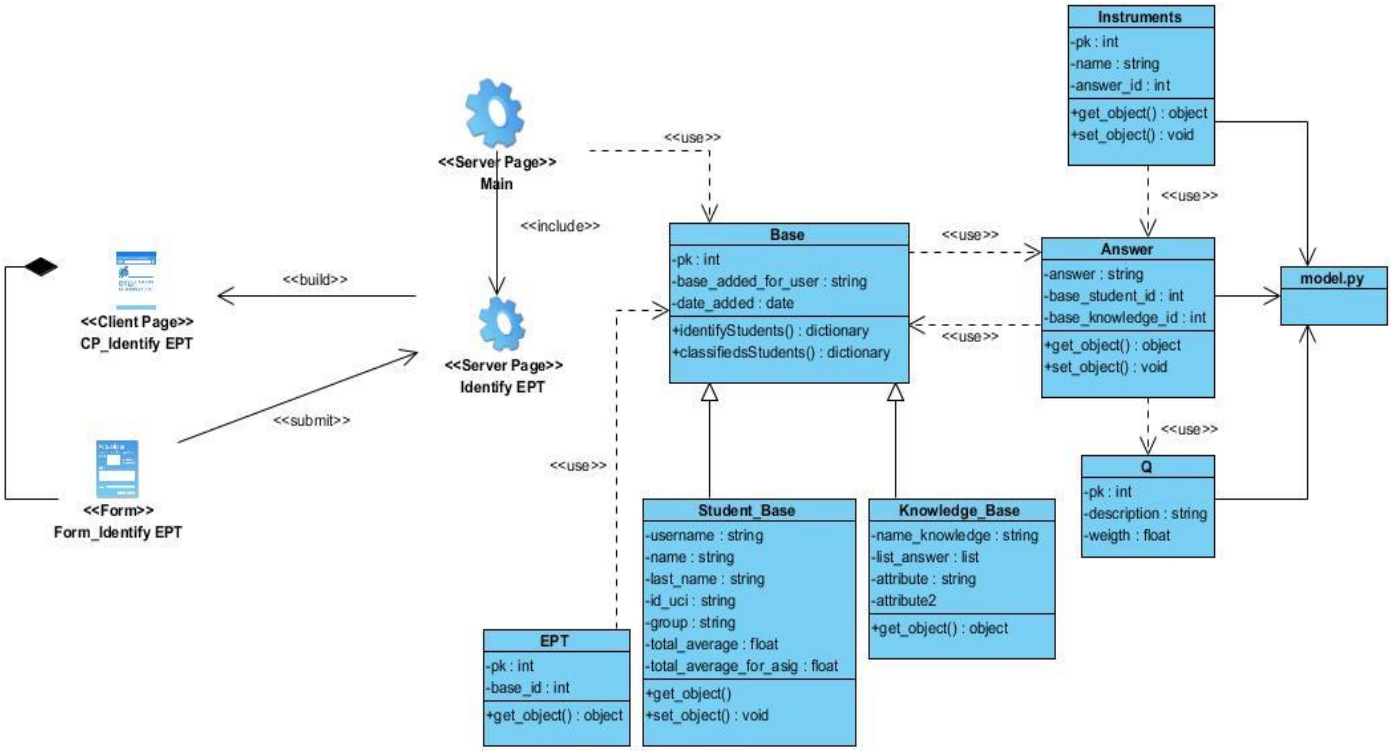

Figura 2. 3 Diagrama de clases con estereotipos web Identificar EPT

CU2.Clasificar EPT:

Nombre	Classify EPT
Actores	Admin
Resumen	El caso de uso inicia cuando termina el proceso de identificación y el administrador selecciona que desea clasificar los EPT identificados.
Complejidad	Alta
Prioridad	Alta
Referencias	RF17, RF23
Precondiciones	Deben existir al menos dos estudiantes identificados como EPT.
Postcondiciones	El sistema muestra los detalles del proceso realizado.
Flujo de eventos	
Flujo básico <Clasificar EPT>	
Actor	**Sistema**

1.	Selecciona la opción Clasificar EPT.	
2.		Muestra un mensaje: “Esta operación puede tardar varios minutos”. Desea continuar.
3.	Selecciona la opción “Aceptar” para continuar el proceso de clasificación.	
4.		Realiza la clasificación de los estudiantes identificados positivamente como EPT haciendo uso del modelo de agrupamiento conceptual.
5.		Muestra un listado con los detalles del proceso culminado.
6.		Termina el caso de uso.
Flujo alterno del flujo básico < Clasificar EPT >Cancelar		
Actor		**Sistema**
3.a	Indica que no desea continuar con la clasificación de los EPT y termina el caso de uso.	
Relaciones	**CU Incluidos**	CU Clasificar EPT.
	CU Extendidos	No aplica.
Requisitos no Funcionales	No aplica.	
Asuntos pendientes	No aplica.	
Prototipo de Interfaz		

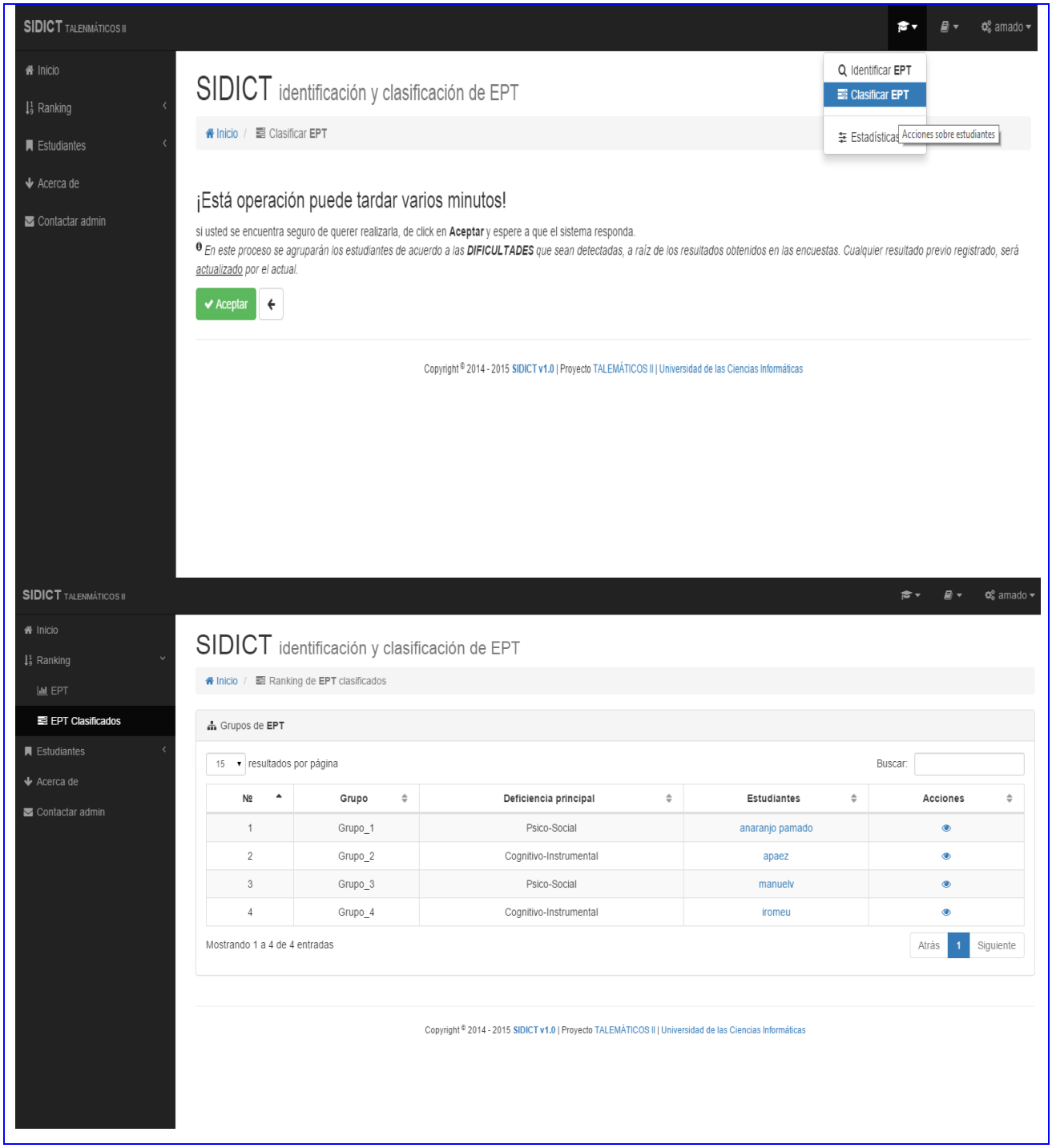

Tabla 2. 3 Descripción del Caso de Uso Clasificar EPT

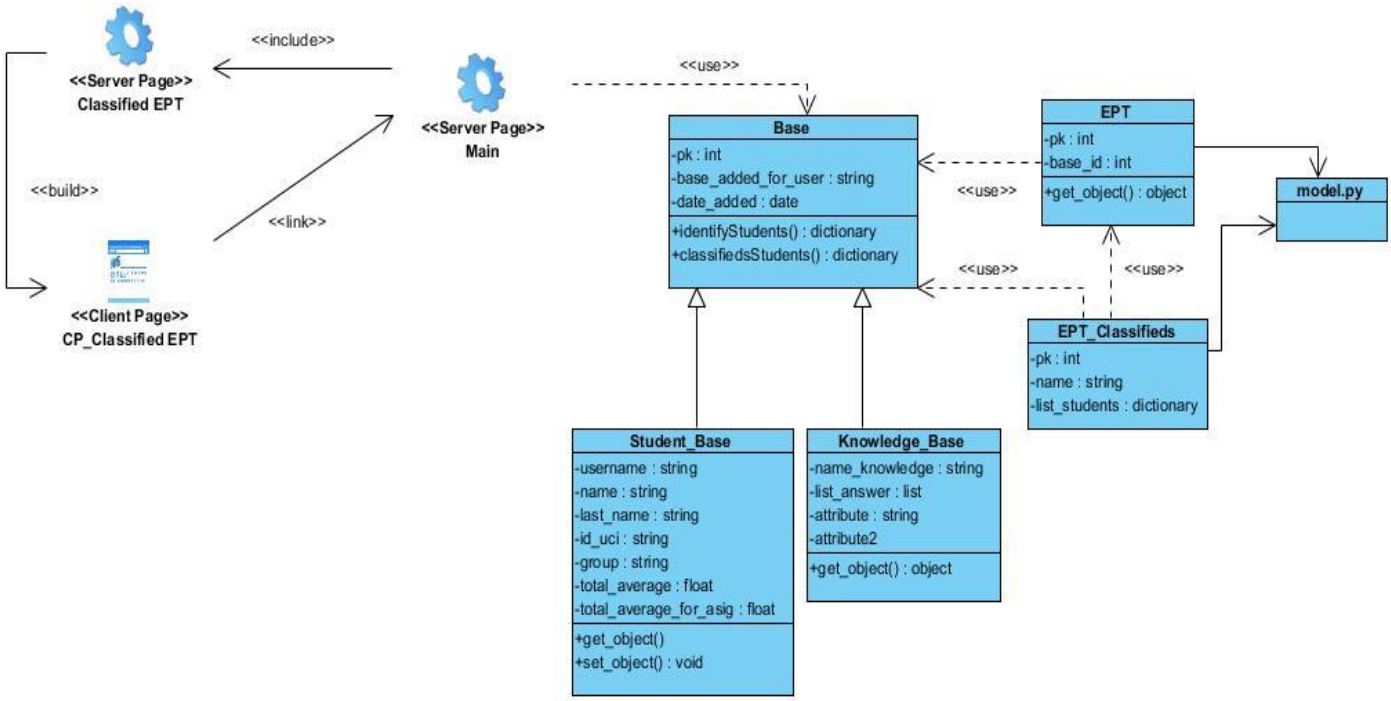

Figura 2. 4 Diagrama de clases con estereotipos web Clasificar EPT

CU3.Gestionar Base de Conocimiento:

Nombre	Manage knowledge base
Actores	Admin
Resumen	El caso de uso inicia cuando el administrador desea insertar, eliminar o ver los detalles de algún caso de la base de conocimiento.
Complejidad	Media
Prioridad	Alta
Referencias	RF9, RF10
Precondiciones	El administrador debe estar autenticado en el sistema.
Postcondiciones	Se insertó o eliminó un caso en el sistema.
Flujo de eventos	
Flujo básico <Gestionar Base de Conocimiento>	

	Actor	**Sistema**
1.	Selecciona una de las siguientes opciones:	

	• Insertar caso (*Ver sección #1*) • Ver detalles del caso (*Ver sección #2*) • Eliminar caso (*Ver sección #3*)	
Sección #1: "Insertar caso"		
Flujo Básico Gestionar Base de Conocimiento		
Actor		**Sistema**
1.1		Ejecuta el CU "Listar casos de la Base de Conocimiento".
1.2		Se listan los casos existentes de la Base de Conocimiento en una interfaz que permite crear nuevos casos.
1.3	Selecciona la opción "Añadir nuevo caso".	
1.4		El sistema muestra una interfaz con las opciones para introducir los datos del caso (nombre del caso, y si se considera EPT), así como las preguntas asociadas a cada Instrumento, que se pueden responder para darle valor caso.
1.5	Introduce los datos para insertar el nuevo caso y selecciona la opción "Guardar".	
1.6		El sistema valida los datos introducidos. Si faltan datos obligatorios ver flujo alterno 1.6a "Faltan datos obligatorios".
1.7		El sistema inserta un nuevo caso en la Base de Conocimiento.
1.8		Termina el caso de uso.

Sección #2: “Ver detalles del caso”		
Flujo Básico Gestionar Base de Conocimiento		
Actor		**Sistema**
2.1		Ejecuta el CU “Listar casos de la Base de Conocimiento”.
2.2		Se listan los casos existentes de la Base de Conocimiento en una interfaz que permite crear nuevos casos.
2.3	Selecciona la opción “Ver detalles del caso”.	
2.4		Muestra los datos asociados al caso seleccionado: • Nombre del caso • Si es o no EPT • Los resultados de las encuestas
2.5		Termina el caso de uso.
Sección #3: “Eliminar caso”		
Flujo Básico Gestionar Base de Conocimiento		
Actor		**Sistema**
3.1		Ejecuta el CU “Listar casos de la Base de Conocimiento”.
3.2		Se listan los casos existentes de la Base de Conocimiento en una interfaz que permite crear nuevos casos.
3.3	Selecciona la opción “Eliminar caso”.	
3.4		Muestra un mensaje: “Está seguro de querer Eliminar el caso de la

		Base de Conocimiento".
3.5	Selecciona la opción "Estoy seguro".	
3.6		Elimina el caso seleccionado de la Base de Conocimiento.
3.7		Ejecuta el CU "Identificar EPT".
3.8		Termina el caso de uso.
Flujos alternos del flujo básico < Gestionar Base de Conocimiento >		
"Guardar y añadir otro" Insertar caso 1.5a		
Actor		**Sistema**
1	Selecciona el botón "Guardar y añadir otro".	
2		Ejecuta el paso 1.6 de la sección "Flujo Básico Gestionar Base de Conocimiento".
3		El sistema inserta un nuevo caso en la Base de Conocimiento.
4		Ejecuta el CU "Añadir nuevo caso".
"Guardar e ir a la base casos" Insertar caso 1.5b		
Actor		**Sistema**
1	Selecciona el botón "Guardar e ir a la base de casos".	
2		Ejecuta el paso 1.6 de la sección "Flujo Básico Gestionar Base de Conocimiento".
3		El sistema inserta un nuevo caso en la Base de Conocimiento.
4		Ejecuta el CU "Listar Base de Conocimiento".
"Cancelar" Insertar caso 1.5c		
Actor		**Sistema**

1	Selecciona el botón “Cancelar”.	
2		Limpia los campos y regresa al paso 1 de la sección “Flujo Básico Gestionar Base de Conocimiento”.
Flujos alternos 1.6a		
Faltan datos obligatorios		
Actor		**Sistema**
1		El sistema marca en rojo los campos requeridos. Regresa al paso 1.4 de la sección “Insertar caso”.
2	Introduce los datos para insertar el caso y selecciona el botón “Guardar”.	
3		Regresa al paso 1.6 de la sección “Insertar caso”.
“Cancelar” Eliminar caso 3.5a		
Actor		**Sistema**
1	Selecciona el botón “Cancelar”.	
2		Ejecuta el CU “Listar Base de Conocimiento”.
Relaciones	**CU Incluidos**	CU Listar Base de Conocimiento, CU Identificar EPT.
	CU Extendidos	No aplica.
Requisitos no Funcionales	No aplica.	
Asuntos pendientes	No aplica.	
Prototipo de Interfaz		

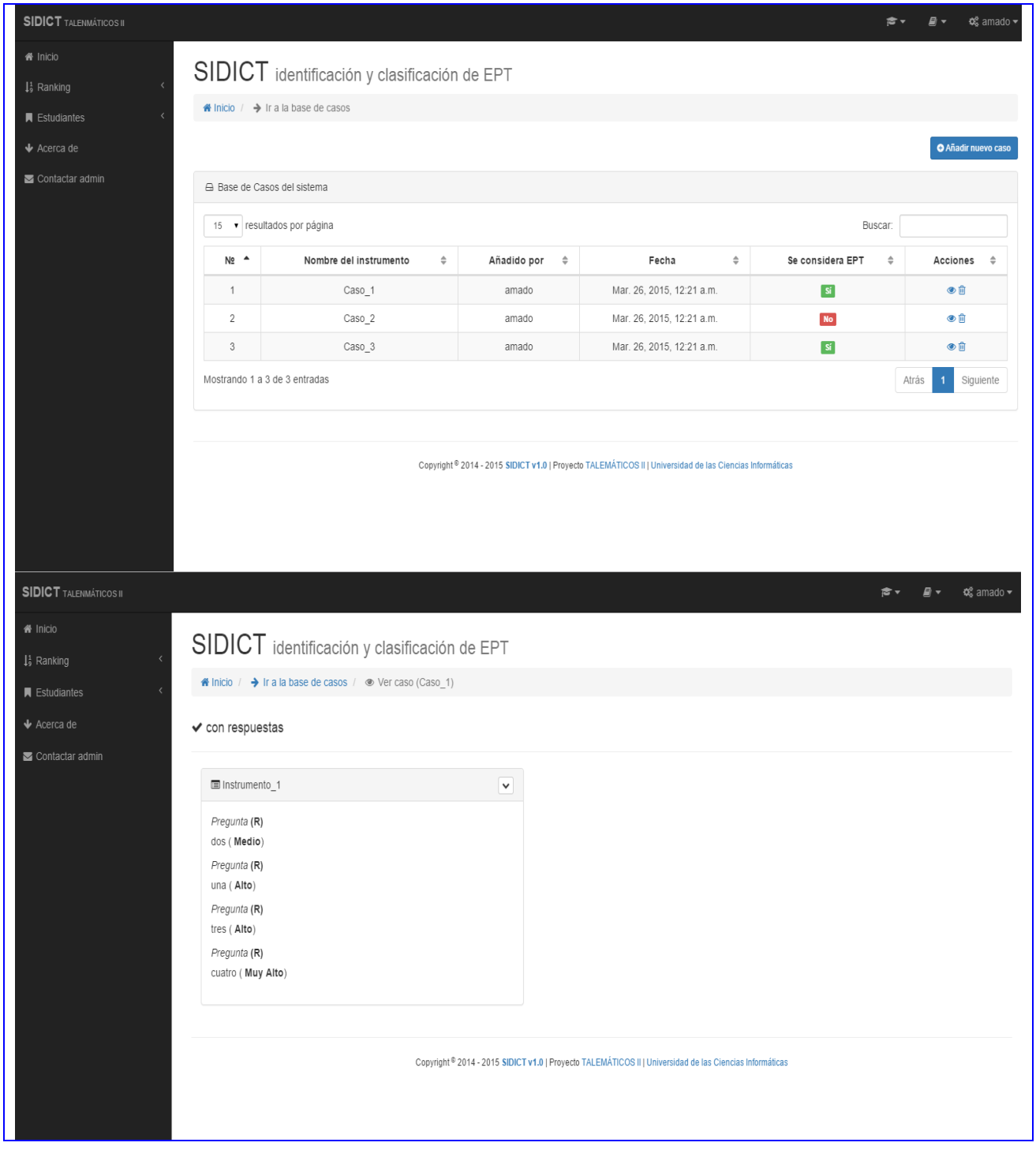
SIDICT TALENMÁTICOS II
amado
Inicio
Ranking
Estudiantes
Acerca de
Contactar admin
SIDICT identificación y clasificación de EPT
Inicio / Ir a la base de casos
Añadir nuevo caso
Base de Casos del sistema
15 resultados por página
Buscar:
№ | Nombre del instrumento | Añadido por | Fecha | Se considera EPT | Acciones
1 | Caso_1 | amado | Mar. 26, 2015, 12:21 a.m. | Si
2 | Caso_2 | amado | Mar. 26, 2015, 12:21 a.m. | No
3 | Caso_3 | amado | Mar. 26, 2015, 12:21 a.m. | Si
Mostrando 1 a 3 de 3 entradas
Atrás 1 Siguiente
Copyright © 2014 - 2015 SIDICT v1.0 | Proyecto TALEMÁTICOS II | Universidad de las Ciencias Informáticas
SIDICT TALENMÁTICOS II
amado
Inicio
Ranking
Estudiantes
Acerca de
Contactar admin
SIDICT identificación y clasificación de EPT
Inicio / Ir a la base de casos / Ver caso (Caso_1)
con respuestas
Instrumento_1
Pregunta (R)
dos (Medio)
Pregunta (R)
una (Alto)
Pregunta (R)
tres (Alto)
Pregunta (R)
cuatro (Muy Alto)
Copyright © 2014 - 2015 SIDICT v1.0 | Proyecto TALEMÁTICOS II | Universidad de las Ciencias Informáticas

Tabla 2. 4 Descripción del Caso de Uso Gestionar Base de Conocimiento

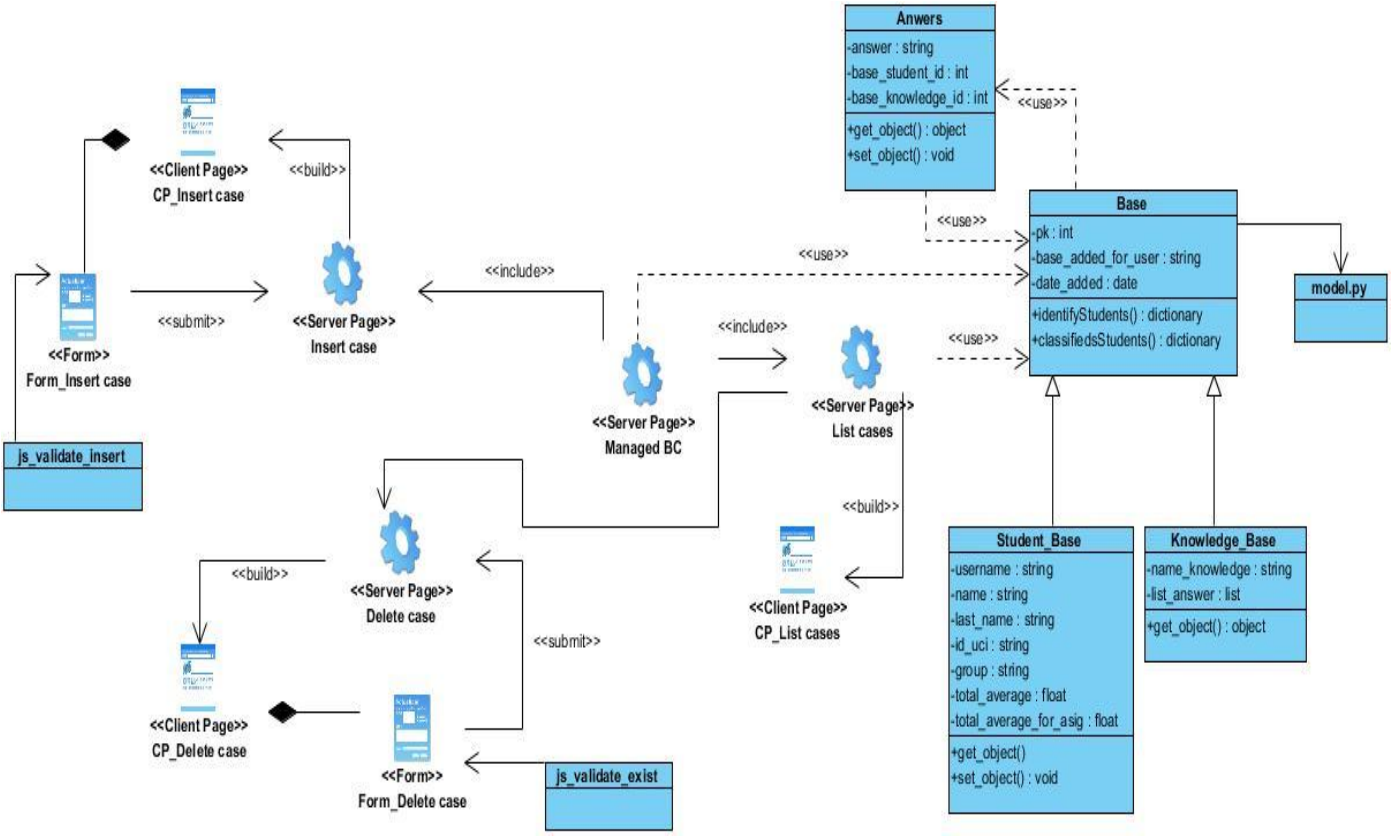

Figura 2. 5 Diagrama de clases con estereotipos web Gestionar Base de Conocimiento

2.5 Descripción de la Arquitectura

Django se basa en el Modelo Vista Controlador (MVC[9]), aunque sus desarrolladores prefieren llamarlo Modelo Plantilla Vista (MTV, por sus siglas en inglés). El controlador pasa a ser la vista y la vista pasa a denominarse plantilla. En Django, una vista describe los datos que se ofrecen al usuario pero no necesariamente su aspecto. Una vista habitualmente delega los datos a una plantilla que describe la forma de presentarlos. Se dice que el controlador de un patrón MVC clásico estaría representado por el propio framework.

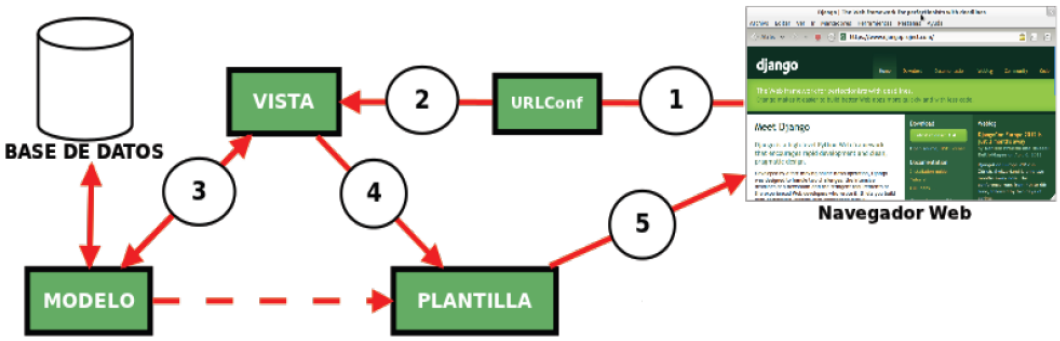

Figura 2. 6 Representación de la arquitectura (Django Project, 2015)

2.6 Patrones de Diseño

"Un patrón es una descripción de un problema y la solución a la que se da un nombre, y que se puede aplicar a nuevos contextos; idealmente, proporciona consejos sobre el modo de aplicarlo en varias circunstancias." (Larman, 1999)

Resulta una buena práctica antes de elaborar los diagramas de interacción asignar correctamente las responsabilidades. Los patrones de diseño constituyen entonces, el auxiliar principal para esta tarea.

A continuación se describen los patrones de diseño, puestos en práctica en el desarrollo del sistema.

2.6.1 Patrones GRASP

[9] MVC (Modelo Vista Controlador). Arquitectura de software que permite la separación de los datos de la aplicación, la interfaz de usuario y la lógica del control en tres componentes distintos.

Los patrones GRASP[10] describen los principios fundamentales de la asignación de responsabilidades a objetos, además constituyen un apoyo para entender el diseño y aplica el razonamiento para el diseño de una forma sistemática, racional y aplicable. (Larman, 1999)

- **Patrón Creador**

Encargado de la creación de instancias. Es uno de los patrones más comunes en un sistema orientado a objetos. Se pone de manifiesto al utilizar las class-based-views[11] de Django

- **Patrón Experto**

Asigna la responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

- **Patrón Bajo Acoplamiento**

Asigna responsabilidades a las clases para mantener bajo acoplamiento. Explica cómo dar soporte a una dependencia escasa y a un aumento de la reutilización.

- **Patrón Alta Cohesión**

Asigna responsabilidades de forma que la cohesión entre las clases siga siendo alta. Explica cómo mantener la complejidad dentro de los límites manejables.

- **Patrón Controlador**

Asigna la responsabilidad del manejo de los eventos del sistema a una clase que represente un sistema global.

2.6.2 Patrones GoF

A principios de los años 90 con la publicación del libro Design Patterns, se establecen veintitrés patrones de diseño GoF[12]. Los patrones GoF se descubren como una forma indispensable de enfrentarse a la programación a raíz del libro “Design Patterns-Elements of Reusable Software” de

[10] GRASP es un acrónimo que significa General Responsibility Asignment Software Patterns (patrones generales de software para asignar responsabilidades)
[11] Las class-based-views son vistas que permiten realizar de forma sencilla, las funcionalidades básicas en casi todos los sistemas (Crear, Actualizar, Eliminar, Listar)
[12] GoF (Gang of Four o Banda de Cuatro)

Erich Gamma, Richard Helm, Ralph Jonson y John Vlissides, a partir de entonces estos patrones son conocidos como los patrones de la pandilla de los cuatro (GoF, Gang of Four). (Pressman, 2002)

Los patrones GoF complementan a los patrones GRASP y en ocasiones se puede encontrar una contraposición entre este tipo de patrones, e incluso, podría inferirse que muchos de los patrones GoF son variantes de los patrones GRASP.

- **Patrón Mediator**

Encargado de manejar la interacción entre los diferentes subsistemas. Realiza el mapeo objeto-relacional a cargo del motor de Django.

2.7 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar. Además sirve de abstracción de la implementación del sistema y es, de ese modo, utilizada como entrada fundamental de las actividades de implementación. (Jacobson, y otros, 2000)

2.7.1 Diagrama de Secuencia

Un diagrama de secuencia muestra una interacción que está organizada como una secuencia temporal. En particular, muestra los objetos que participan en la interacción mediante sus líneas de vida y mediante los mensajes que se intercambian, organizados en forma de una secuencia temporal.

A continuación se muestran los diagramas de secuencia correspondiente a los principales CU del sistema:

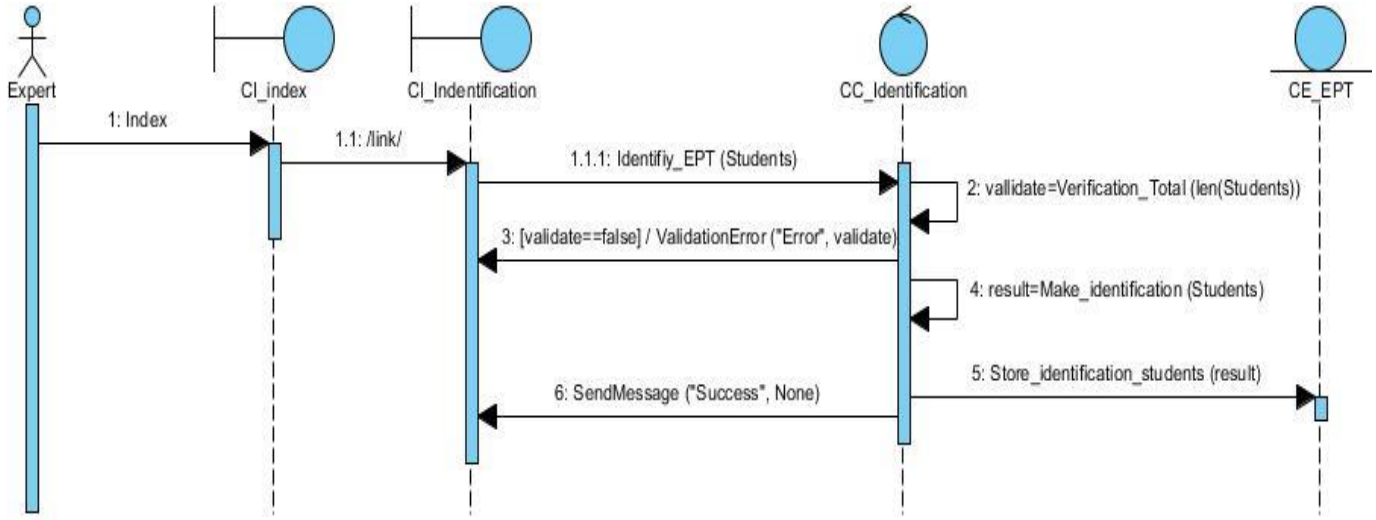

Figura 2. 7 Diagrama de Secuencia del CU Identificar EPT

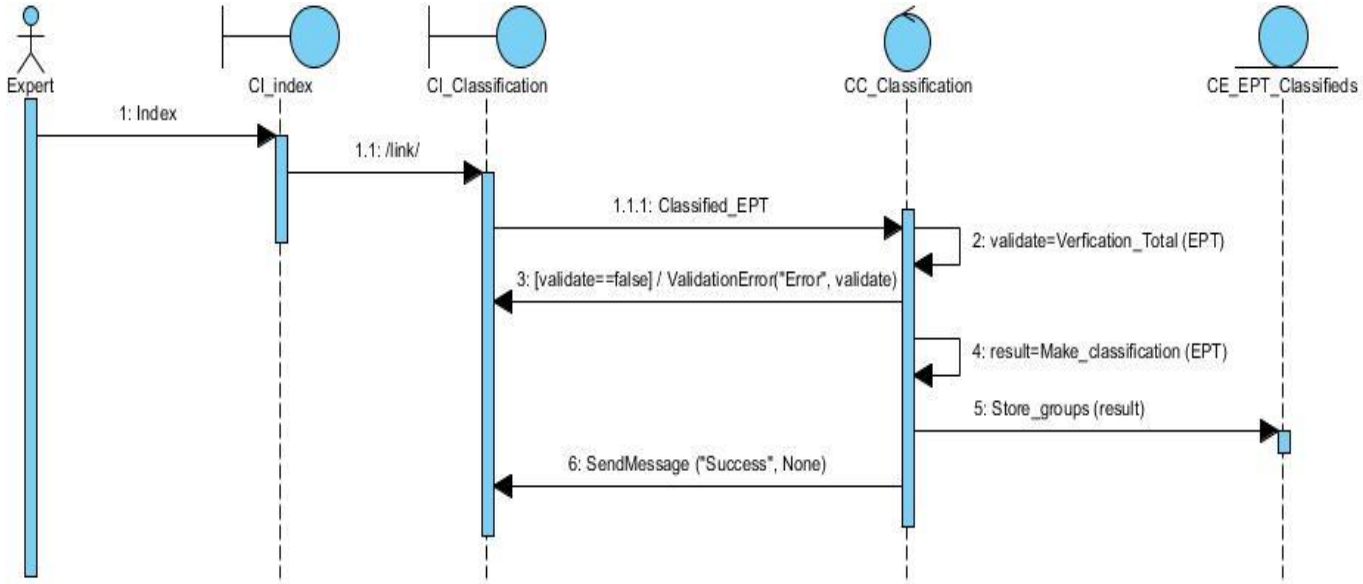

Figura 2. 8 Diagrama de Secuencia del CU Clasificar EPT

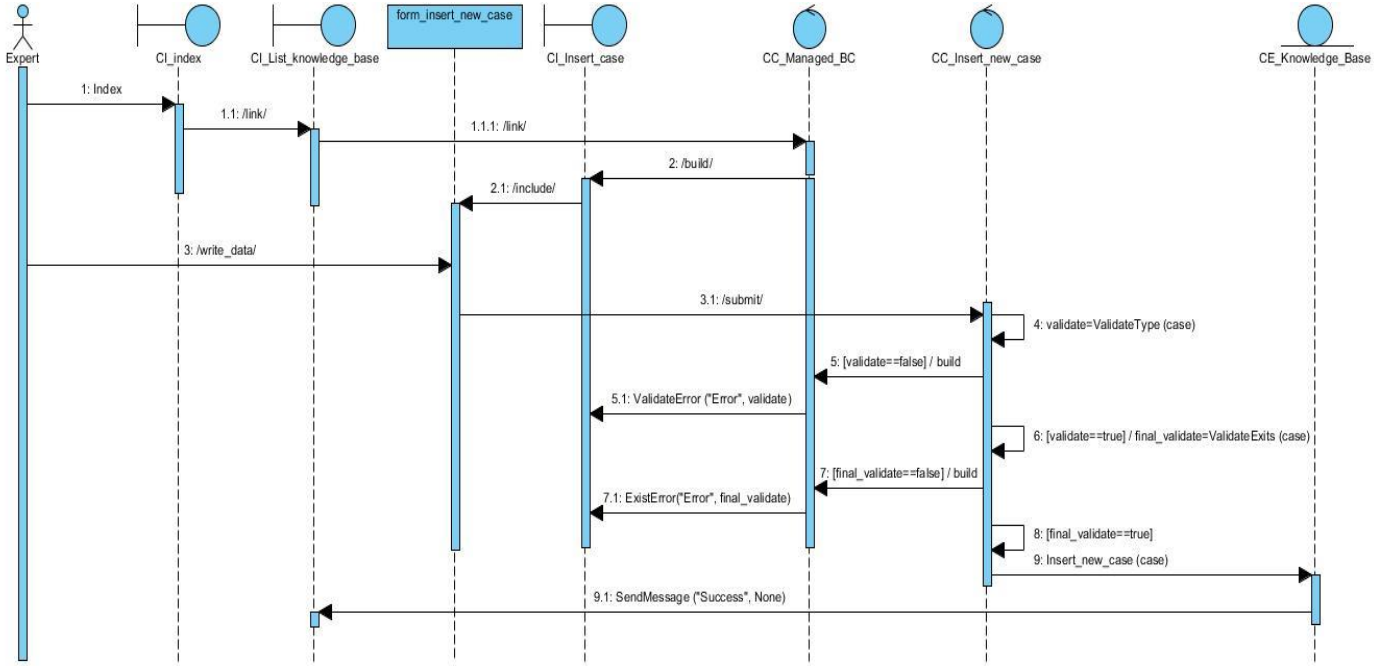

Figura 2. 9 Diagrama de Secuencia del CU Insertar nuevo caso

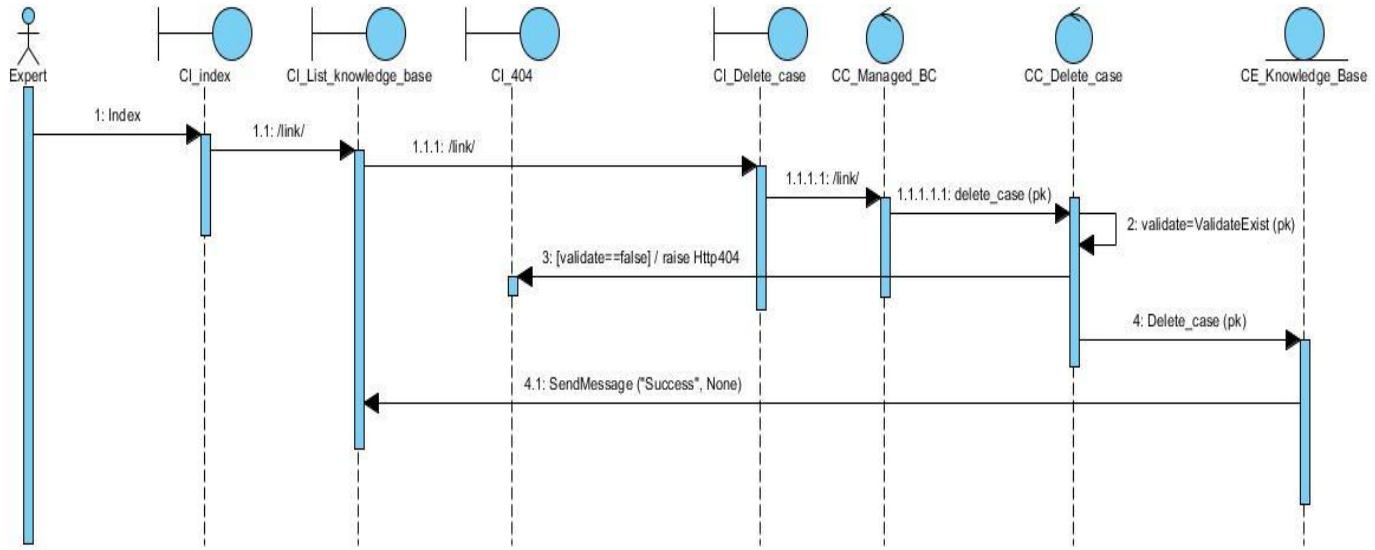

Figura 2. 10 Diagrama de Secuencia del CU Eliminar caso

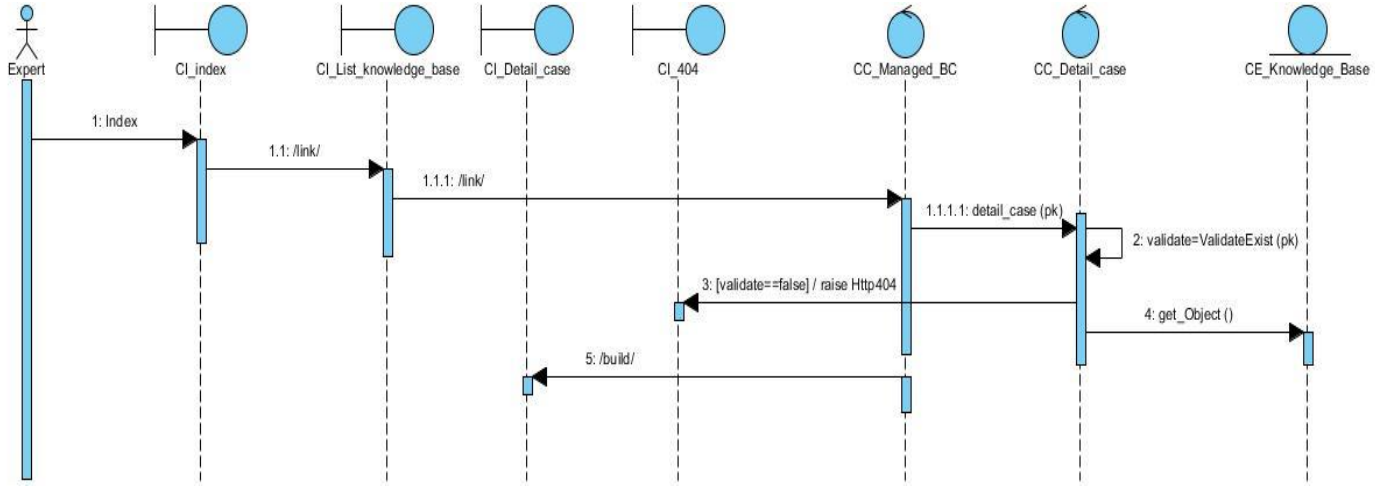

Figura 2. 11 Diagrama de Secuencia del CU Ver detalles del caso

2.8 Modelo de datos del sistema

El modelo de datos es un lenguaje orientado a describir las bases de datos. Un modelo de datos permite describir la estructura de datos de la base y su relación, las restricciones de integridad y las operaciones de manipulación de datos. Los modelos de datos aportan la base conceptual para diseñar aplicaciones que hacen un uso intensivo de datos, así como la base formal para las herramientas y técnicas empleadas en el desarrollo y uso de sistemas de información. (Universidad Autónoma de Barcelona, 2010)

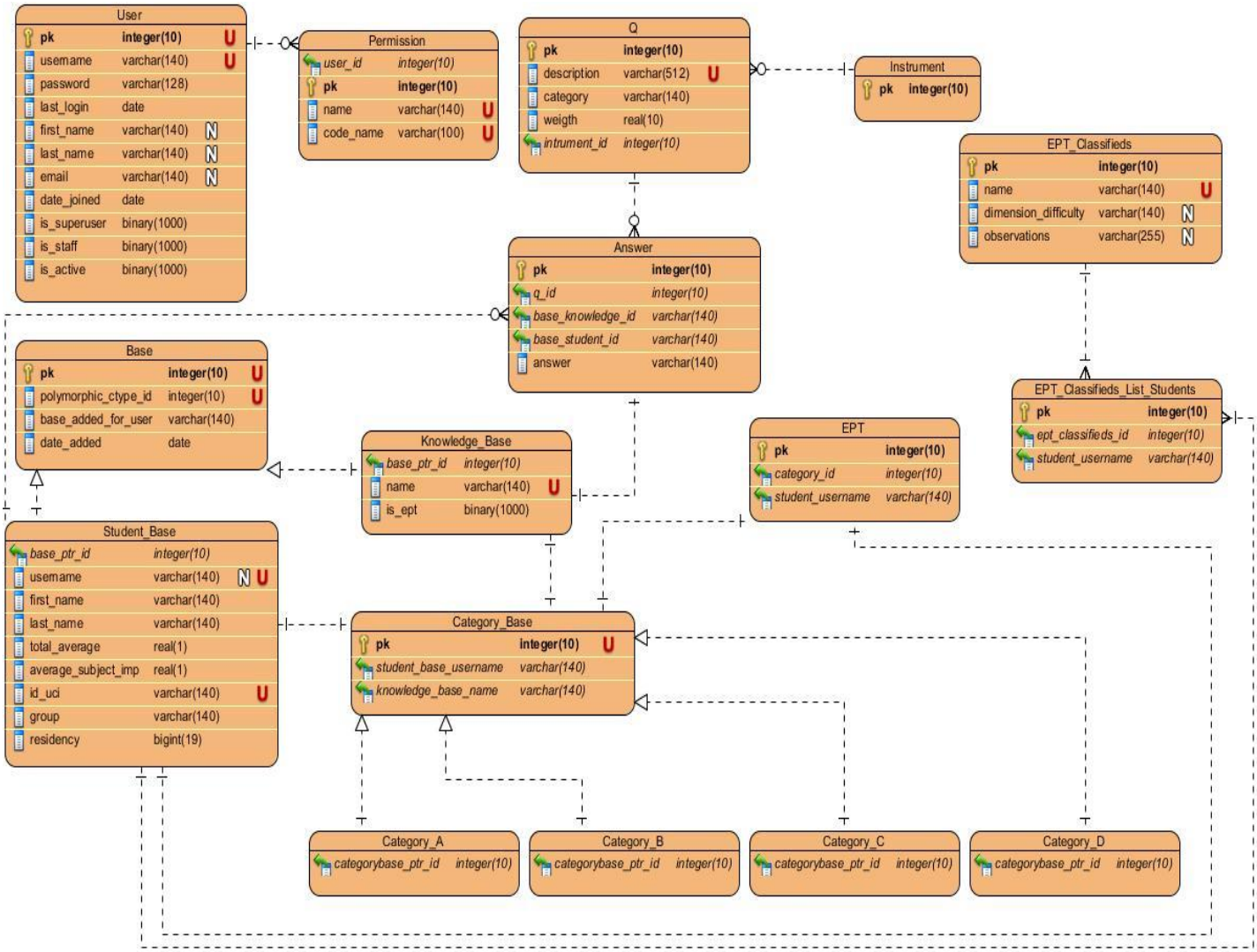

Figura 2. 12 Modelo de datos del sistema

2.9 Conclusiones parciales

En el presente capítulo se dio cumplimiento al segundo objetivo específico sobre la implementación de la herramienta informática. Se expuso la propuesta de solución del sistema que se desea desarrollar. Se abarcaron de forma más detalles aspectos principales de los modelos de IA seleccionados para la construcción de la herramienta, así como la descripción de la base de conocimientos. Se describieron los procesos de identificación y clasificación de EPT, esclareciendo de esta forma que significa cada proceso en el contexto del proyecto Talenmáticos y que técnicas se

utilizaran para llevar a cabo estas tareas. Se realizó el modelo de dominio mostrándose todas las clases conceptuales utilizadas, además se realizó el diagrama de casos de uso. Para guiar la creación de la herramienta se definió la arquitectura y los patrones de diseño, así como las descripciones, diagramas de clases y secuencia de los casos de uso de mayor relevancia en el contexto del problema planteado. Por último se realizó el modelo Entidad-Relación para representar la estructura de la base de datos utilizada.

Capítulo 3: Implementación y Prueba

3.1 Introducción

El presente capítulo describe cómo los elementos del modelo de diseño son implementados. Además se lleva a cabo el proceso de pruebas del sistema, en el cual se hace referencia a las diferentes pruebas realizadas. Por último se realizaron los casos de prueba, obteniéndose un resultado de las pruebas realizadas.

3.2 Diagrama de Componentes

El diagrama de componentes ilustra los componentes del software que serán usados para construir el sistema. Estos pueden ser construidos para el modelo de clase y escritos para satisfacer los requisitos del nuevo sistema, también muestran la relación entre los componentes del software, sus dependencias, comunicaciones, localización y otras condiciones, examinan y controlan las dependencias entre componentes o interfaces de los componentes y representa una parte modular, desplegable y reutilizables del sistema.

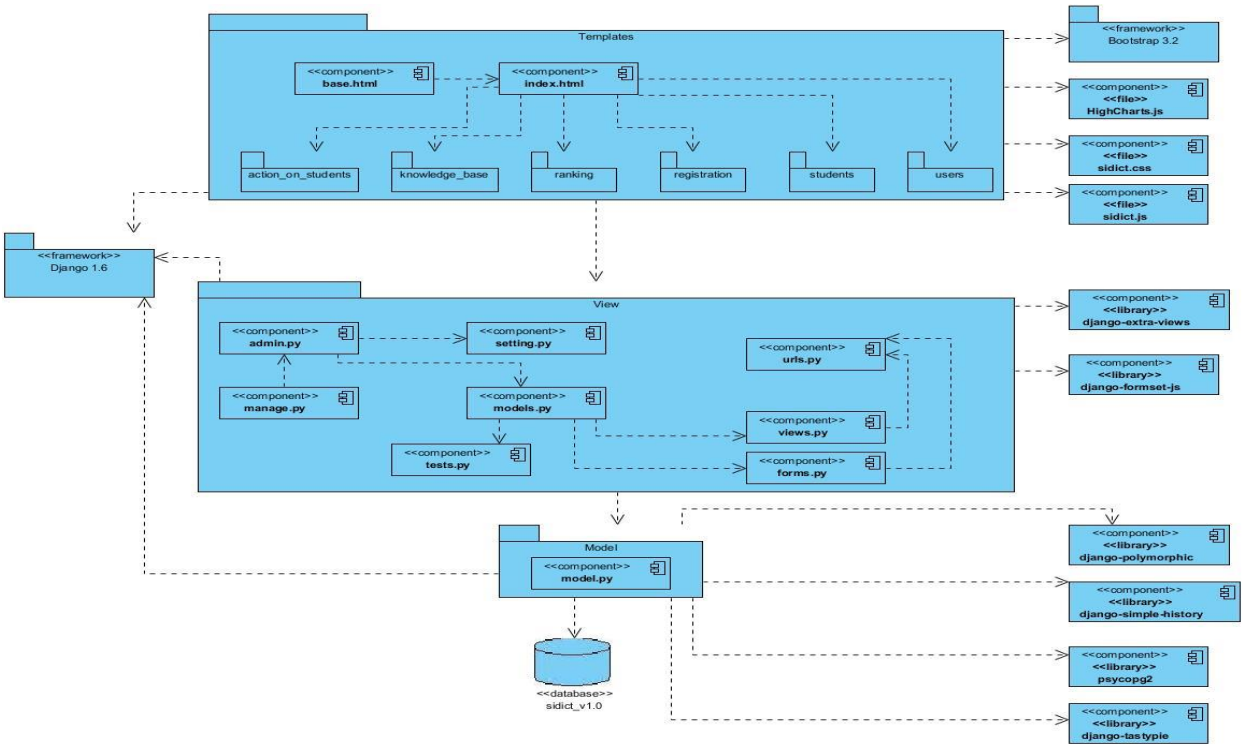

Figura 3. 1 Diagrama de Componente

3.3 Diagrama de Despliegue

Un diagrama de despliegue modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan en esos nodos. (Spark Systems, 2010)

Estos tipos de diagramas son más limitados que los de componentes, ya que solo representan la estructura del sistema en producción y no el sistema en su fase de desarrollo.

A continuación se muestra el diagrama de despliegue del sistema, el cual está compuesto por un nodo Client PC que representa las estaciones de trabajo de los usuarios que utilizarán la aplicación, un nodo BD sidict_v1.0 para representar la base de datos del sistema y por último el nodo Server Nginx que representa al servidor de aplicaciones webs.

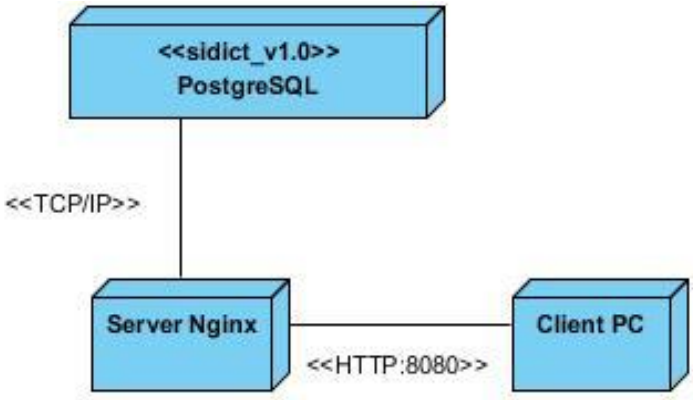

Figura 3. 2 Diagrama de Despliegue

3.4 Pruebas

Con el desarrollo de software ha de ir aparejado algún proceso o actividad que garantice su calidad. Las pruebas de software son un elemento indispensable para ello. Pueden identificar diferencias entre el resultado esperado y el obtenido, así como verificar que se satisfacen los requisitos del sistema desarrollado. *"La prueba no puede asegurar la ausencia de defectos: sólo puede demostrar que existen defectos en el software".* (Tabares, 2011)

En el desarrollo de aplicaciones web el proceso de probar el sistema de una manera correcta supone un gran esfuerzo. Esto está dado por el hecho de que los requisitos del sistema están sujetos a

cambios constantes, lo que implica un gran número de versiones del sistema y con ello la aparición de nuevos errores.

Este es el motivo por el que la automatización de pruebas es una recomendación, aunque no una obligación, útil para crear un entorno de desarrollo satisfactorio. Las pruebas automatizadas permiten garantizar que los cambios no introducen incompatibilidades en el funcionamiento del sistema.

Django incorpora su propio framework de pruebas automatizadas (PyUnit). Las mismas se escriben y ejecutan desde un archivo que se crea desde el inicio del proyecto llamado test.py. Lo que diferencia a los test automatizados, es que el trabajo de hacer las pruebas lo hace el sistema por uno. Cuando se crean los test, a medida que se hacen cambios en la aplicación, se puede verificar que el código todavía funciona como estaba originalmente pensado, sin tener que usar tiempo para hacer pruebas de forma manual.

3.4.1 Pruebas Unitarias

Las pruebas unitarias son las encargadas de detectar errores en los datos, y en la lógica. Estas aseguran que un único componente de la aplicación produzca una salida correcta para una determinada entrada.

Las pruebas unitarias se realizan porque: (Mellado, 2004)

- Aseguran la calidad del código entregado, pero no asegura detectar todos los errores, por tanto las pruebas de integración y aceptación siguen siendo necesarias.
- Permiten encontrar errores en los inicios del desarrollo, teniendo en cuenta que mientras más temprano se corrijan los errores, menos costará corregirlos.
- Facilitan que el programador cambie el código para mejorar su estructura (refactorización), puesto que permiten hacer pruebas sobre los cambios y asegurarse de que no han introducido errores.
- Las pruebas funcionales se hacen más sencillas pues la mayoría de los aspectos individuales de cada unidad ya están probados. De este modo, las pruebas funcionales se centran solo en verificar la correcta cooperación de las distintas unidades y los funcionamientos generales del programa.

A continuación se muestran un fragmento de código de prueba al modelo Estudiante:

```
class Student_Base_Test(TestCase):
    def setUp(self):
        self.user = User.objects.create(username="amado", password="amado")

    def test_insert_student(self):
        student = Student_Base.objects.create(base_added_for_user=self.user, username="anaranjo", first_name="Amado",
                                              last_name="Naranjo Comas", total_average=4.105, average_subject_imp=4.0,
                                              id_uci="E132654", group="F1452")
        self.assertEquals(student.base_added_for_user, self.user)
        self.assertEquals(student.username, "anaranjo")
        self.assertEquals(student.first_name, "Amado")
        self.assertEquals(student.last_name, "Naranjo Comas")
        self.assertEquals(student.total_average, 4.105)
        self.assertEquals(student.average_subject_imp, 4.0)
        self.assertEquals(student.id_uci, "E132654")
        self.assertEquals(student.group, "F1452")
        self.assertEquals(Student_Base.objects.count(), 1)
        self.assertEquals(Base.objects.count(), 1)
        self.user.username = "manuel"
        self.user.save()
        self.assertEquals(student.base_added_for_user, self.user)

    def test_views(self):
        self.assertTrue(self, self.client.login(username="amado", password="amado"))
        response = self.client.get(reverse('list_students'))
        self.assertEquals(response.status_code, 302)

        response = self.client.get(reverse('add_new_student'))
        self.assertEquals(response.status_code, 302)

```

Figura 3. 3 Fragmento de código de prueba al modelo Estudiante

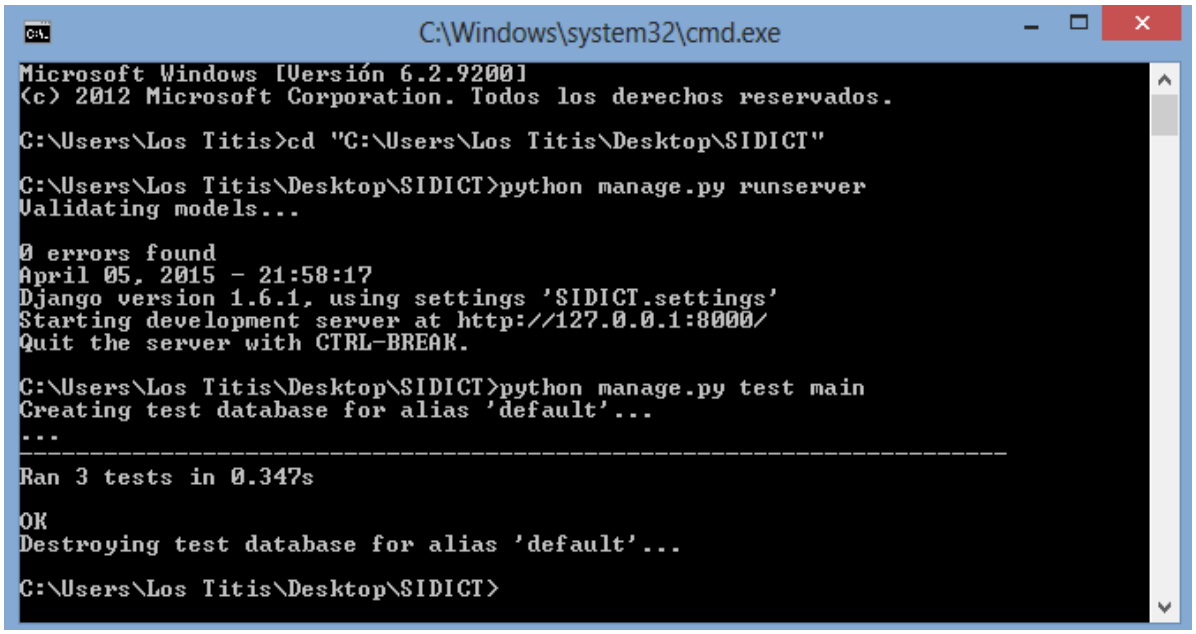

Figura 3. 4 Resultado de la prueba realizada al modelo Estudiante

3.4.2 Pruebas de Aceptación

Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra. (Pressman, 2002). La prueba de software, es un elemento crítico para la garantía de la calidad del mismo y representa una revisión final de las especificaciones del diseño y de la codificación. La prueba está enfocada principalmente en la evaluación y determinación de la calidad del producto. La técnica seleccionada es la prueba de caja negra, la cual se centra en los requisitos funcionales de la aplicación, sin entrar en detalles del funcionamiento interno de la misma, se desarrollan sobre la interfaz visual del software y se utiliza para detectar errores en la interfaz, funciones incorrectas, errores de salida y problemas con el acceso a datos. (Van Wyk, 2009)

Para confeccionar los casos de prueba de caja negra existen distintos criterios; algunos de ellos son: (Pressman, 2002)

- Técnica de la Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

3.4.2.1 Partición de Equivalencia

La partición equivalente es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico. (Pressman, 2002)

3.4.3 Diseño de Casos de Prueba

CU Gestionar Base de Conocimiento

Descripción general

El CU inicia cuando un experto autenticado como súper-usuario carga la BC del sistema y selecciona una de las opciones, adicionar nuevo caso, ver detalles del caso o eliminar caso.

Condiciones de ejecución

- El usuario debe estar autenticado como súper-usuario en el sistema.
- Debe existir al menos un Instrumento insertado en el sistema.

Sección a probar CU Gestionar Base de Conocimiento

Escenarios de la sección	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Añadir nuevo caso.	Se inserta un nuevo caso en la BC del sistema.	El sistema inserta un nuevo caso en la BC y muestra un mensaje del proceso satisfactorio.	**1.** Selecciona la opción **"Añadir nuevo caso"**. **2.** Se muestran los datos a introducir para la creación del nuevo caso. **3.** Introduce los datos del nuevo caso y selecciona la opción **"Guardar"**. **4.** El sistema verifica que todos los campos requeridos se han llenado. **5.** Si los datos fueron introducidos correctamente, el sistema guarda los datos del nuevo caso en la BC.
EC 1.2 Ver detalles del caso	Se muestran los detalles del caso seleccionado.	El sistema muestra una lista con los detalles del caso seleccionado.	**1.** Selecciona la opción **"Ver detalles del caso"**. **2.** El sistema verifica que el caso exista en la BC. **3.** Muestra los datos del caso seleccionado.
EC 1.3 Eliminar caso	Se elimina un caso seleccionado de la BC.	El sistema elimina un caso de la BC.	**1.** Selecciona la opción **"Eliminar caso"**. **2.** El sistema verifica que el caso seleccionado exista en la BC. **3.** Selecciona la opción **"Estoy seguro"**.

Tabla 3. 1 Diseño de casos de prueba del CU Gestionar Base de Conocimiento

Casos de prueba del CU Gestionar Base de Conocimiento

Escenario	Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba
EC 1.1	El usuario entra los datos válidos para añadir un nuevo caso a la BC del sistema.		El sistema muestra un mensaje de confirmación indicado que el proceso ha concluido de forma satisfactoria.	El sistema muestra el mensaje “Se añadió un caso satisfactoriamente a la BC”.
		El usuario no entra los datos requeridos para el nuevo caso.	El sistema alerta, marcando en rojo los campos incorrectos.	El sistema marca en rojo los campos incorrectos, alertando al usuario del error cometido en ese campo.
		El usuario inserta un caso que ya existe en la BC del sistema.	El sistema alerta, mostrando un mensaje de error.	El sistema muestra el mensaje “El caso que se desea insertar ya existe en la BC”.
EC 1.2	El usuario selecciona un caso válido para ver sus datos.		El sistema lista los datos del caso seleccionado.	El sistema muestra lista con los datos del caso seleccionado.
		El usuario selecciona un caso incorrecto (no existe) de la BC.	El sistema lanza un error Http404.	El sistema muestra una página de error 404.
EC1.3	El usuario selecciona la opción “Eliminar” de un caso.		El sistema elimina satisfactoriamente el caso de la BC.	El sistema muestra el mensaje “Se eliminó satisfactoriamente el caso de la BC”.
		El usuario selecciona la opción “Eliminar” para un caso inválido (no existe) en la BC.	El sistema lanza un error Http404.	El sistema muestra una página de error 404.

Tabla 3. 2 Casos de prueba del CU Gestionar Base de Conocimiento

CU Identificar EPT

Descripción general

El CU inicia cuando un experto autentificado como súper-usuario selecciona la opción Identificar EPT.

Condiciones de ejecución

- El usuario debe estar autenticado como súper-usuario en el sistema.
- Debe existir al menos un estudiante insertado en la base de datos del sistema.
- Debe existir al menos un caso insertado en la BC del sistema.

Sección a probar CU Identificar EPT

Escenarios de la sección	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Identificar EPT	Se identifican los estudiantes existentes en la Base de Datos del sistema en las categorías A (se consideran EPT), B (cercano a ser EPT), C (no es EPT), D (no existen casos semejantes para él).	El sistema muestra una vista con las cuatro categorías, listando por defecto la categoría A.	**1.** Selecciona la opción **"Identificar EPT"**. **2.** Se muestra una página de confirmación de la acción, alertando que la operación puede tardar varios minutos. **3.** Se listan las cuatro categorías, a partir de la inferencia realizada.

Tabla 3. 3 Diseño de casos de prueba del CU Identificar EPT

Casos de prueba del CU Identificar EPT

Escenario	Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba
EC 2.1	El usuario registra al menos un estudiante y un caso en el sistema.		El sistema muestra una vista con opciones para poder ver el resultado de las cuatro categoría, listando por defecto la categoría A.	El sistema muestra una vista con el resultado de la identificación, listando por defecto los estudiantes que pertenecen a la categoría A.
		El usuario no registra ningún estudiante, ni casos en el sistema.	El sistema alerta, mostrando un mensaje con enlaces a las vistas “Añadir nuevo estudiante” y “Añadir nuevo caso”.	El sistema muestra una página de error, con enlaces a las vistas para añadir nuevos estudiantes y casos al sistema.
		El usuario registra solamente estudiantes en el sistema.	El sistema alerta, la NO existencia de casos en la BC y muestra un enlace a la vista “Añadir nuevo	El sistema muestra una página de error, alertando que no se tienen registrados casos y por ende no se puede realizar

			caso”.	la identificación, proveyendo un enlace para poder adicionar nuevos casos a la BC.
		El usuario registra solamente casos en la BC.	El sistema alerta, la NO existencia de estudiantes en la base de datos del sistema y muestra un enlace a la vista “Añadir nuevo estudiante”.	El sistema muestra una página de error, alertando que no se tienen estudiantes registrados y por ende no se puede realizar la identificación de EPT, proveyendo un enlace para poder adicionar nuevo estudiantes a la base de datos del sistema.

Tabla 3. 4 Casos de prueba del CU Identificar EPT

CU Clasificar EPT

Descripción general

El CU inicia cuando un experto autentificado como súper-usuario selecciona la opción Clasificar EPT.

Condiciones de ejecución

- El usuario debe estar autenticado como súper-usuario en el sistema.
- Debe existir al menos tres estudiantes insertados en la base de datos del sistema.
- Debe existir al menos un caso insertado en la BC del sistema.
- Se debe haber realizado la Identificación de EPT, arrojando como resultado al menos tres estudiantes como EPT.

Sección a probar CU Clasificar EPT

Escenarios de la sección	Descripción	Respuesta del sistema	Flujo central
EC 3.1 Clasificar EPT	Se identifican los estudiantes	El sistema lista los grupos formados en el proceso de	**1.** Selecciona la opción **"Clasificar EPT"**.

	existentes en la Base de Datos del sistema y se procede a realizar la clasificación de los estudiantes identificados satisfactoriamente, redireccionando a una vista que lista los grupos formados.	clasificación de estudiantes identificados como EPT.	**2.** Se muestra una página de confirmación de la acción, alertando que la operación puede tardar varios minutos. **3.** Se listan los grupos formado en el proceso de clasificación.

Tabla 3. 5 Diseño de casos de prueba del CU Clasificar EPT

Casos de prueba del CU Identificar EPT

Escenario	Clases válidas	Clases inválidas	Resultado esperado	Resultado de la prueba
EC 3.1	Se obtienen en el proceso de identificación al menos tres EPT.		El sistema lista los grupos formados en el proceso de clasificación y brinda opciones para ver las características de cada grupo.	El sistema muestra una vista, listando los grupos formados en el proceso de clasificación y alertando para cada grupo formado, donde están las principales deficiencias, así como las opciones para ver a mayor detalle las características del grupo.
		El usuario no realiza previamente la identificación de EPT.	El sistema alerta, mostrando un mensaje con enlace a la vista de "Identificar EPT".	El sistema muestra una página de error, con enlace a la vista para identificar los EPT.
		El usuario obtiene como resultado del proceso de identificación menos de tres estudiantes.	El sistema alerta, que no se puede realizar la identificación con menos de tres estudiantes identificados como EPT.	El sistema muestra una página de error, alertando que el proceso de clasificación no se puede llevar a cabo, por tener menos de tres estudiantes identificados como EPT.

Tabla 3. 6 Casos de prueba del CU Clasificar EPT

3.4.4 Resultado de Pruebas

Se realizaron tres iteraciones de pruebas, detectándose de forma general 18 no conformidades, el siguiente gráfico muestra el resultado obtenido:

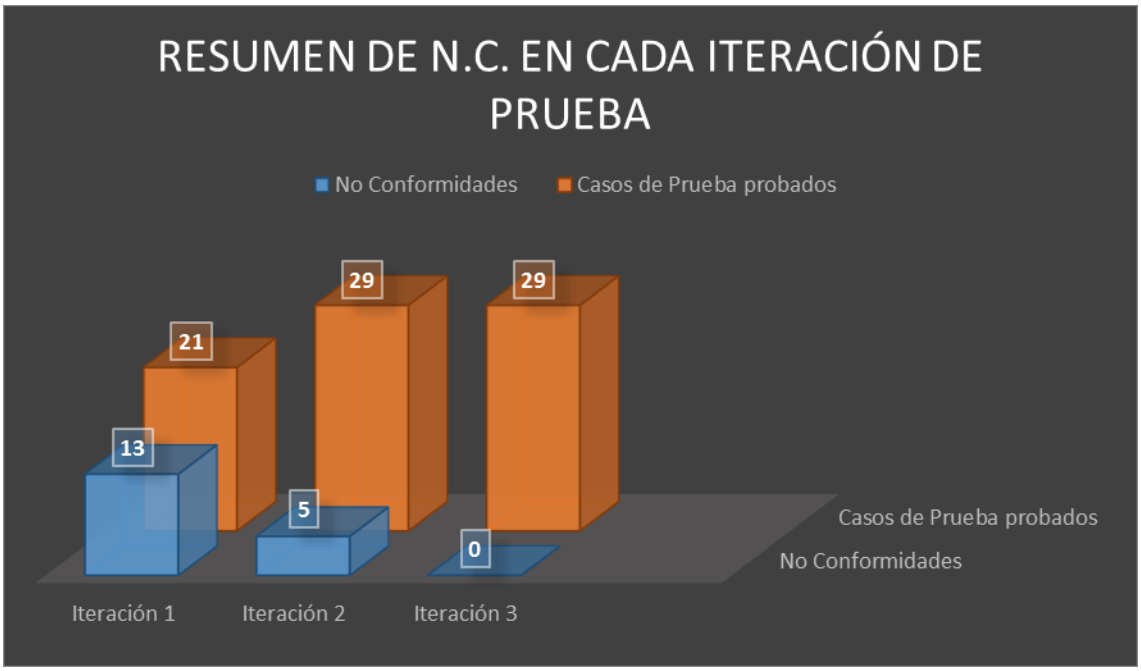

Figura 3. 5 No conformidades detectadas en cada iteración de prueba

Iteración 1.	Se probaron 21 casos de prueba y se detectaron 13 No Conformidades
Iteración 2.	Se probaron 29 casos de prueba y se detectaron 5 No Conformidades
Iteración 3.	Se probaron 29 casos de prueba y se detectaron 0 No Conformidades

3.5 Conclusiones parciales

En este capítulo se dio cumplimiento al tercer objetivo específico propuesto. Se realizó el diagrama de componente y despliegue, las pruebas unitarias utilizando el framework de pruebas de Django (PyUnit). Además se utilizó el método de caja negra en la validación de los requisitos funcionales del software utilizando la técnica de partición de equivalencia. Las pruebas realizadas en el presente capítulo permitieron a los desarrolladores corregir errores detectados en la herramienta desarrollada y sentaron las bases para futuras pruebas y mejoras de alguna funcionalidad.

Conclusiones Generales

Al culminar el desarrollo del presente trabajo se concluye que:

- Se desarrolló el SIDICT en correspondencia con los fundamentos teóricos y los requisitos metodológicos establecidos en el proyecto Talenmáticos, con respecto a la identificación de EPT, como parte del modelo para la atención educativa a los EPT en Informática.
- El SIDICT es capaz de apoyar los procesos de toma de decisiones que se llevan a cabo en el proyecto Talenmáticos, con respecto a la identificación de EPT, pues a partir de determinados resultados obtenidos por encuestas, se pueden predecir posibles EPT en Informática, además se establecen agrupaciones dentro de los EPT que permiten trabajar sobre las principales deficiencias detectadas en la identificación.

Recomendaciones

Como futuras mejoras al SIDICT los autores recomiendan:

- Mejora de la interfaz gráfica. Establecer un entorno más sencillo de manejar para el usuario.
- Dotar al sistema de una herramienta de generación de informes para que el usuario final pueda extraer en PDF u otro formato la relación de los EPT identificados.

Bibliografía

1. **Bello Pérez, Rafael y otros. (2002):** *Aplicaciones de la Inteligencia Artificial.* Jalisco: s.n., 970-27-0177-5.
2. **Carpineto, C. & Romano, G. (1996).** *A lattice conceptual clustering system and its application to browsing retrieval.* Machine Learning. 24(2): pp. 95-122.
3. **Castellanos, D. y María D. Córdova (2003).** *Hacia una comprensión de la inteligencia. En Talento: Estrategias para su desarrollo*. La Habana. Editorial Pueblo y Educación.
4. **Comunidad SPARK SYSTEMS. [En línea] [Citado el: 8 de marzo de 2010.]** http://www.sparxsystems.com.ar/resources/tutorial/uml2_deploymentdiagram.html.
5. **Cordero Morales, D., Ruiz Constanten, Y., Torres Rubio, Y. (2013)**. *Sistema de Razonamiento Basado en Casos para la identificación de riesgos de software*. Revista Cubana de Ciencias Informáticas. Vol. 7, No.2, pp. 101-110, La Habana, Cuba. ISSN 1994-1536
6. **Django Project, enero 22, (2015)**. http://www.djangoproject.com
7. **Fisher, D. H. (1987).** *Knowledge Acquisition Via Incremental Conceptual Clustering.* Machine Learning. 2(2): pp.139-172.
8. **Gálvez, Daniel. (1998)**. *Curso de Sistemas Basado en el Conocimiento*. Villa Clara, Cuba. Universidad Central “Marta Abreu” de las Villas, T83-T825.
9. **Gutiérrez, I; Bello, R. y Tellería, A. (2002).** *Un Sistema Basado en Casos para la toma de decisiones en condiciónes de incertidumbre.* Revista de Investigación Operacional, Vol.23 No. 2, pp. (103-121).
10. **Gutiérrez Rodríguez, Andrés E. y otros. (2012).** *Algoritmos de agrupamiento basado en patrones utilizando árboles de decisión no supervisados*. Coordinacion de Ciencias Computacionales INAOE, Reporte Técnico No. CCC-12-002, Puebla México**.**
11. **Isasi Viñuela, Pedro y Galván León, Inés M. (2003).** *Redes de Neuronas Artificiales. Un enfoque práctico.* Editorial Pearson Prentice Hall, ISBN 84-205-4045-0
12. **Jacobson, I., Booch, G., and Rumbaugh, J. (2000)**. *El Proceso Unificado de Desarrollo de Software*. s.l. : Addison Wesley, 2000.
13. **Jiawei, Han & Micheline, Kamber. (2006)** *Data Mining: Concepts and Techniques.* Illinois. EEUU: Morgan Kaufmann Publishers,. ISBN 978-1-55860-901-3, ISSN 1-55860-901-6.

14. **Larman, Craig. 1999.** *UML y Patrones Introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall. Vol. 2da Edición.
15. **Lebowitz, M. (1987).** *Experiments with Incremental Concept Formation: UNIMEM*. Machine Learning. 2(2):pp. 103-138.
16. **López Pino, José L**., **(2011).** *Servidores web más usados.* En: [en línea]. 2009. [Accedido Diciembre 2011]. Disponible desde: http://lopezpino.es/2010/07/30/servidores-web-mas-usados/.
17. **Lorenzo García, R., (2010)**. *El Talento ¿se hereda o se adquiere?* La Habana. Editorial Científico Técnica.
18. **Martín del Brío, Bonifacio y Sanz Molina, Alfredo. (2001)**. *Redes Neuronales y Sistemas Difusos*. 2da edición ampliada y revisada, Editorial RA-MA. Zaragoza, España.
19. **Martínez-Trinidad, J. F. y otros. (2001).** *LC: A Conceptual Clustering Algorithm***.** Proceedings of the Second International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer-Verlag: pp. 117-127.
20. **Martínez Sánchez, N; García Lorenzo, MM. y García Valdivia, ZZ. (2009).** *Modelo para diseñar sistemas de enseñanza-aprendizaje inteligentes utilizando el razonamiento basado en casos.* Revista Avances en Sistemas e Informática, Vol.6 No.3, p.(72).
21. **Mellado, Julio. (2004).** *Estrategia de pruebas de Líneas de Productos de Sistemas de Tiempo Real especificados con diagramas de estado jerárquico.* Universidad Politécnica de Madrid, España.
22. **Menéndez Pérez, Jorge S., y otros. (2012)**. *Definición de talento y de talento informático en el marco del Proyecto Talenmático*. [Digital] Buenos Aires: Centro de Estudios Internacionales para el Desarrollo.
23. **Michalski, R. S. & Stepp, R. E (1979).** *Conceptual Clustering: A Theoretical Foundation and a Method for Partitioning Data into Conjunctive Concepts***.** Textes des exposes du Seminaire organise par 'Institute de Recherche d'Informatique et d'Automatique (IRIA): pp. 254-294.
24. **Michalski, R. S. & Stepp, R. E. (1981).** *Concept-based Clustering versus Numerical Taxonomy*. Invited paper submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.
25. **Naranjo Comas, A., Páez Valdés, A., Soria Ramírez, Jorge A. (2014)**. *Algoritmos Conceptuales. Toma de decisiones y selección de talentos en el Juez en Línea Caribeño*. XVII Convención de Ingeniería y Arquitectura CIIA (Cujae), Noviembre (24-28). La Habana, Cuba. ISBN 978-959-261-467-3

26. **Ortiz Torres, Emilio, Aguilera Pupo, Eleanne y González Pérez del Villar, Ana María., (2010)**. *Los estilos de aprendizaje, la superdotación intelectual y el talento en estudiantes universitarios*. 5, Holguín: Centro de Estudios sobre las Ciencias de la Educación Superior (CECES), Vol. 5.
27. **Patón, Dr. Eduardo Fernández Medina. (2006-2007)**. *INGENIERÍA DEL SOFTWARE I* Cuarto Curso. [En línea] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf**.**
28. **Peña Ayala, Alejandro. (2006).** *Sistemas basados en Conocimiento: Una Base para su Concepción y Desarrollo.* Dirección de Publicaciones del Instituto Politécnico Nacional de México, 1era edición. ISBN 970-94797-4-1
29. **Pérez Suárez, Airel y Medina Pagola, José E., (2014).** *Algoritmos de agrupamiento conceptual de objetos*. CENATAV, RNPS No. 2143, ISSN 2072-6260
30. **Petrovski, A. (1986)**: *Psicología General. Manual didáctico para los institutos de pedagogía*. Editorial Progreso. Moscú. Tercera Edición.
31. **Piñero Pérez, Pedro Y. (2005).** *Un modelo para el aprendizaje y la clasificación automática basado en técnicas de softcomputing.* Tesis en opción del grado científico de "Doctor en Ciencias Técnicas". Universidad Central "Marta Abreu" de las Villas.
32. **Pons Porrata, A. (2002). RGC**: *A new conceptual clustering algorithm for mixed incomplete data sets*. In Mathematical and Computer Modelling. pp. 1375-1385.
33. **Pons Ponrrata, A. (2003). LEX:** *Un nuevo algoritmo para el cálculo de los testores típicos.* Revista Ciencias Matemáticas. Cuba**.**
34. **PostgreSQL, enero 22, (2015)**. http://www.postgresql.org
35. **Pressman, Roger S. (2002).** *Ingeniería de Software. Un enfoque práctico*. Madrid : Mc Graw Hil.
36. **Python, enero 21, (2015)**. http://www.python.org
37. **Python Software Foundation, enero 21, (2015)**. http://www.python.org/psf/
38. **Ralambondrainy, H. (1995).** *A conceptual version of the K-means algorithm*. Pattern Recognition. Lett. Vol.16, No.11, pp.1147-1157.
39. **Real Academia Española, RAE. 2001.** *Diccionario de la lengua española. Vigésiam segunda edición.* Real Academia Española. Madrid,España : s.n., 2001.

40. **Reyes González, Yunia y Martínez Sanchéz, Natalia (2014).** *La toma de decisiones en los Sistemas Tutoriales Inteligentes utilizando el agrupamiento conceptual.* Revista Cubana de Ciencias Informáticas. Vol.8 No.Especial UCIENCIA 2014, pp. (104-116), ISSN 2227-1899.
41. **Ruiz Shulcloper, José (1988).** *Problemas de clasificación de objetos y selección de variables en medios difusos: tres proyectos de tesis.* Revista del Seminario de Enseñanza y Titulación. Año IV No.21, UNAM México,pp.(41-68).
42. **Ruiz Shulcloper, José (1990).** *Modelos matemáticos de Reconocimiento de Patrones.* Editorial UCLV, Santa Clara, Cuba.
43. **Ruiz Shulcloper, José y Lazo, M. (1991).** *K-tesores primos generalizados.* Revista de Ciencias Físico-Técnicas y Matemáticas, No.9, pp.(17-55).
44. **Ruiz Shulcloper, José; E. Alba y Lazo, M. (1995).** *Introducción a la teoría de testores.* Serie Verde, No.50, CINVESTAV-IPN. México.
45. **Ruiz Shulcloper, José y Lazo Cortés, Manuel. (1995).** *Introducción al Reconocimiento de Patrones: enfoque lógico combinatorio*. Serie Verde No. 51, CINVESTAV-IPN. México.
46. **Ruiz Shulcloper, José. (2013):** *Acerca del surgimiento del Reconocimiento de Patrones en Cuba.* Revista Cuba de Ciencias Informáticas. Vol. 7, No.2, pp 30-47, La Habana Cuba. ISSN 1994-1536
47. **Russel, Norving (1995).** *IA Bible Artificial Intelligent: A Modern Aproach***.**
48. **Sommerville, I. (2005)**. *Ingeniería del software*. España: Pearson Educación.
49. **Storti, Guillermo, Gladys Ríos and Campodónico, Gabriel (2007).** [En línea] 2007. [Citado el: mayo 16, 2013.] http://www.belgrano.esc.edu.ar/matestudio/carpeta_de_access_introduccion.pdf.
50. **Tabares, Marta Silvia (2011).** *Introducción a las Pruebas de Software.* 2011.
51. **Universidad Autónoma de Barcelona. [En línea] [Citado el: 9 de junio de 2010]** http://elies.rediris.es/elies9/4-2.htm
52. **Van Wyk, K. (2009).** *Build Security In.* Recuperado el 9 de Junio de 2012, de Homeland Security: https://buildsecurityin.us-cert.gov/bsi/articles/tools/black-box/261-BSI.html
53. **Vergara Panzeri, M. (s/a):** *Necesidades educativas de los Talentosos y Dotados*. Buenos Aires. Centro para el Desarrollo del Alto Potencial. Formato digital.
54. **Visual Paradigm. [En línea] noviembre 26, (2011)**. [Citado el: diciembre 16, 2014.] http://www.visual-paradigm.com

Glosario de Siglas

Abreviaturas	Significado
SIDICT	Sistema Inteligente De Identificación y Clasificación de Talentos
PEA	Proceso de Enseñanza Aprendizaje
EPT	Estudiantes Potencialmente Talentosos
SBC	Sistemas Basados en Conocimiento
RBC	Razonamiento Basado en Casos
AC	Algoritmos de agrupamiento Conceptual
BC	Base de Casos o Base de Conocimiento
RP	Reconocimiento de Patrones
IDE	Entorno Integrado de Desarrollo
RUP	Rational Unified Process o Proceso Unificado de Desarrollo.
UML	Lenguaje Unificado de Modelado
CASE	Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora

Tabla 4. 1 Glosario de Siglas