UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Módulo Teoría de Grafos del sistema interactivo y experimental para el proceso de enseñanza-aprendizaje de la Matemática Discreta.

Trabajo de diploma para optar por el título

de Ingeniero en Ciencias Informáticas

Autor: Osmel Mojena Dubet

Tutores:

MSc. Alién García Hernández

Ing. José Ángel Álvarez Abraira

Ing. Yarileidis Barcena Calzado

La Habana, Julio de 2017

El mayor placer de la vida es hacer lo que la gente dice que no puedes

Walter Bagehot

Agradecimientos

Hoy es uno de los días más importantes de mi vida, hoy que me gradúo de ingeniero en ciencias informáticas, no puedo dejar de agradecer a todas esas personas que hicieron posible este sueño, por tal motivo agradezco.

A mi madre que hay sido faro y guía en todo momento de mi vida, por su apoyo incondicional y estar siempre pendiente de mis estudios y estar hoy aquí celebrado conmigo este triunfo.

A mi padre y hermanos que siempre me han apoyado y han estado pendientes en todo momento de este largo camino.

A mi maestra de primaria Anisia, que hasta el día de hoy que me gradúo en la universidad ha estado pendiente de todos estos años de estudio.

A mi novia Laura, por todo su apoyo, comprensión y paciencia durante todo el tiempo que estuve inmerso en este largo proceso.

A mi suegra Sandra y a Ernesto, que me han brindado todo su apoyo y se han convertido en parte de mi familia.

A mis primos Lisbet y Rolevis por su dedicación, apoyo y estar siempre pendiente de mí.

A mis abuelos y tíos, que me han ayudado mucho a lo largo de mi vida y hoy están más orgullosos que nunca de esta victoria.

A mi tutor Alién, por confiar en mí y darme la oportunidad de realizar esta tesis y estar siempre dispuesto a ayudarme en todo momento.

A mi tutor José, quien estuvo estos largos cuatro meses al lado mío aclarando todas mis dudas sin importar horarios ni fines de semana.
A mi tutora Yaris, quien desde segundo año ha estado pendiente cada uno de mis pasos y hasta el día de hoy me ha apoyado en todos mis proyectos.
A una persona que hoy tengo el placer de conocer en persona, Sarín, quien me ha apoyado siempre y ha estado pendiente de mis estudios.
A todos mis familiares y amigos que de uno forma u otra contribuyeron a la realización de este trabajo de diploma que hoy se convierte en una nueva victoria.
A mi amigo Ortelio que desde primer año me ha apoyado en mis estudios y el responsable de que hoy sea ingeniero.
A todos mis compañeros de aula quienes han sido mi familia en la UCI durante estos cinco años, en especial a Aimet, Lázaro y Jorge.
A Rachel quien ha sido como una hermana para mí, quien me ha brindado su apoyo, ánimo y confianza, tanto en la tesis como en mi vida personal.
A mis alumnos del pre que de igual manera siempre estuvieron pendientes de mi tesis, en especial a Melissa, Dianelis, Nancy, Chabeli, Adrianita, Jennifer y Miguel Ángel.
En general a todas las personas que hicieron posible que hoy este aquí graduándome de Ingeniero en Ciencias Informáticas. A todos mis más sinceros agradecimientos.

Dedicatoria

Que hoy me forme como ingeniero es parte del trabajo y dedicación de muchas personas a las cuales hoy les dedico esta nueva victoria.

A mi madre que con su demonstración de una madre ejemplar me ha enseñado a no desfallecer ni rendirme ante nada y siempre perseverar a través de sus sabios consejos.

A mi padre y hermanos que siempre estarán ahí para mí en todo momento.

A mi novia Laura quien ha sido víctima del sacrificio, tiempo y dedicación que conlleva realizar esta tarea.

A mis abuelos, tíos y primos quienes deben estar impacientes esperándome para celebrar conmigo este triunfo.

A mis tutores, quienes hoy festejan conmigo es triunfo que también es de ellos.

A todos mis familiares, amigos y profesores que creyeron en mí y en este proyecto que hoy se hace realidad.

A todas las personas que hoy están aquí conmigo que también son protagonistas de este triunfo.

Declaración jurada de autoría

Declaro por este medio que yo Osmel Mojena Dubet, con carnet de identidad 93021201149 ser el autor principal del trabajo titulado “Módulo de Teoría de Grafos del sistema interactivo y experimental para el proceso de enseñanza-aprendizaje de la Matemática Discreta”, y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Osmel Mojena Dubet

MSc. Alién García Hernández

Ing. José Ángel Álvarez Abraira

Ing. Yarileidis Barcena Calzado

Resumen

En la actualidad la incorporación de las Tecnologías de la Información y las Comunicaciones (TIC) al proceso de enseñanza-aprendizaje ha servido de apoyo en la formación de estudiantes en todos los niveles. Dentro de los recursos que contribuyen a ello se encuentran los objetos de aprendizaje, los que pueden ser creados basándose en las especificaciones de las herramientas *E-Learning*. La presente investigación tiene planteada como objetivo desarrollar un módulo de Teoría de grafos integrable al sistema web interactivo y experimental para el proceso de enseñanza-aprendizaje de Matemática Discreta que permita la creación de objetos de aprendizaje y aplicar estrategias de gamificación basándose en las especificaciones *E-Learning*. Para lograrlo se analizaron las herramientas informáticas con características similares a la solución propuesta, asimismo se hizo un estudio de los métodos y tecnologías más factibles a utilizar para lograr un correcto diseño del módulo a través del análisis de las metodologías existentes que rigen el desarrollo de software. Como resultado se obtuvo el módulo de Teoría de Grafos que contribuirá al proceso de enseñanza-aprendizaje de la Matemática Discreta.

Palabras clave: *E-Learning*, formación, gamificación, grafos, metodologías, módulo, software.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Las matemáticas son fundamentales para el desarrollo intelectual de la sociedad, ayudan al pensamiento lógico, a razonar ordenadamente y a tener una mente preparada para la crítica y la abstracción. Las matemáticas configuran actitudes y valores en los estudiantes, pues garantizan una solidez en sus fundamentos, seguridad en los procedimientos y confianza en los resultados obtenidos (Guadalupe, Becerra, y Noriega Paltan, 2016).

Según plantea Martín (2016) las matemáticas son un bello lenguaje que se basa en la parte abstracta para inferir en las diversas ramas del conocimiento. Este lenguaje permite abordar diferentes modelos como representación de diversos fenómenos de estudio, sin embargo, no se queda como herramienta de aplicación o como soporte, es también un maravilloso paraíso que podemos abordar en pro de mantener a esta ciencia.

Por otro lado, las matemáticas son también un pilar fundamental en el mundo de la informática, casi todo lo relacionado con la informática necesita matemáticas, por ejemplo, cuando se hacen programas avanzados muchas veces se requieren usar algoritmos matemáticos para que el programa funcione mejor, también el llamado lenguaje de máquina está basado en las matemáticas, que son los números binarios (Bosch y Gascón, 2013).

Existe una estrecha relación entre matemática y la informática. De hecho, la informática surge de la mano de la matemática, con las primeras máquinas de calcular (como precursor del ordenador digital) inventada en 1642 por el matemático francés B. Pascal. También en el siglo XIX el matemático e inventor británico Charles Babbage inventó una serie de máquinas, como la máquina diferencial, diseñadas para solucionar problemas matemáticos complejos. En el siglo XX comenzaron a construirse los primeros ordenadores analógicos y recién en el período de la segunda guerra mundial surge el primer ordenador digital (Dubinsky, 2013).

Otra rama de la matemática relacionada con la informática es la Matemática Discreta (MD), la cual es la encargada del estudio de los conjuntos discretos: finitos o infinitos numerables, en otras palabras, la matemática discreta estudia estructuras cuyos elementos pueden contarse uno por uno separadamente, sin dar lugar a números decimales ni procesos infinitos. Es decir, los procesos en matemática discreta son finitos y contables. La matemática discreta es la base de todo lo relacionado con los procesos digitales, y por tanto, constituyen una parte fundamental en las ciencias informáticas (Rocca, et al., 2016).

Introducción

Dentro de la MD se encuentra el tema Teoría de Grafos, su estudio es importante tanto para las matemáticas como para la teoría de la computación. En esta última disciplina todo es manejado a través de los grafos que son estructuras discretas que constan de puntos y de líneas que se conectan entre sí. A través de los grafos se pueden estudiar las estructuras de redes en Internet, redes neuronales, análisis de los diferentes algoritmos ya existentes, manejo de las estructuras de datos, la creación de los diferentes sistemas operativos manejados hoy en día, entre otras aplicaciones (Hernández y Movilla, 2010).

La asignatura Matemática Discreta se estudia en el primer año de la carrera Ingeniería en Ciencias Informáticas, en la Universidad de las Ciencias Informáticas (UCI), es la base para obtener un pensamiento lógico elevado y permite obtener grandes niveles de abstracción que son fundamentales para otras asignaturas del currículo. La Teoría de Grafos es uno de los temas más extensos de la asignatura Matemática Discreta II por lo que necesita de un estudio profundo y continuo para un mejor entendimiento y razonamiento de los contenidos, con el objetivo de ocasionar menores dificultades en el proceso de aprendizaje de los estudiantes.

La UCI cuenta con un Entorno Virtual de Enseñanza y Aprendizaje (EVEA) que posibilita la auto preparación de los estudiantes apoyándose en las Tecnologías de la Información y las Comunicaciones (TIC), en este entorno virtual se pueden encontrar contenidos y recursos de todas las asignaturas del plan de estudios de la carrera Ingeniería en Ciencias Informáticas, donde se encuentra la Matemática Discreta.

Al realizar un análisis de los recursos de la asignatura MD en el EVEA se percibe que en dichos espacios solo se encuentran documentos, la mayoría de ellos en formato ".pdf", lo que desaprovecha las ventajas que brindan las TIC en cuanto a la utilización, por ejemplo, de Objetos de Aprendizaje como parte de los elementos que pudieran ayudar en la asimilación de los contenidos con un nivel mayor de dificultad.

Además, luego del análisis realizado a los recursos de MD, relacionados con la Teoría de Grafos, disponibles en el EVEA y del estado actual del proceso de enseñanza-aprendizaje de la asignatura en la UCI se pudieron detectar las siguientes deficiencias:

- Los materiales que se encuentran en el espacio de la asignatura en el EVEA-UCI solo proporcionan orientación sin permitir la experimentación en los contenidos de la asignatura.
- Existe dentro del programa de la asignatura una pobre variedad de ejercicios y tareas que limitan la asimilación adecuada de los contenidos.

- El aprovechamiento de las potencialidades TIC en el proceso de enseñanza-aprendizaje (PEA) de la MD es insuficiente, lo que aleja al estudiante de Ingeniería en Ciencias Informáticas del objeto de su profesión.
- Es insuficiente la existencia de recursos que apoyen el PEA de la MD para aquellos contenidos de más difícil y lenta asimilación, que tienen la posibilidad de ser representados de manera gráfica, permitiendo tener un grado de interactividad elevado que permita la experimentación.
- La interacción estudiante-profesor, en el proceso de enseñanza-aprendizaje de la Teoría de Grafos, se centra en actividades presenciales y no se aprovecha de esta manera un aprendizaje ubicuo.
- Existen insuficiencias en la autoevaluación de los estudiantes, los cuáles se limitan a la realización de los ejercicios orientados por sus profesores.

Con el objetivo de resolver esta problemática, en el curso 2016-2017, se desarrolló un sistema interactivo y experimental como apoyo al proceso de enseñanza-aprendizaje de la MD para la Ingeniería en Ciencias Informáticas. Este sistema, que obtuvo premio en el concurso nacional de computación, cuenta con las siguientes limitantes:

- No posibilita la enseñanza de la Teoría de Grafos, rama esencial en el ingeniero que se forma.
- No posibilita estrategias para el uso de la gamificación[1].

Por todo lo anteriormente planteado se determina el siguiente **problema de la investigación:**

¿Cómo contribuir a la autoevaluación y la interacción estudiante-profesor en el proceso de la enseñanza aprendizaje de la Teoría de Grafos mediante el sistema interactivo experimental?

Siendo identificado como **objeto de estudio:** el proceso de enseñanza-aprendizaje de la Teoría de Grafos.

Para dar solución al problema antes descrito se tiene como **objetivo general**: desarrollar el módulo Teoría de Grafos del sistema Web interactivo y experimental para contribuir a la autoevaluación y la interacción estudiante-profesor en el proceso de enseñanza-aprendizaje de la Matemática Discreta.

Para darle cumplimiento al objetivo planteado se responderán las siguientes **preguntas científicas:**

[1] Técnica de aprendizaje novedosa para motivar a los estudiantes a aprender a través de los juegos.

1. ¿Cuáles son los supuestos teóricos que sustentan el desarrollo y la utilización del módulo de Teoría de Grafos del Sistema Web interactivo y experimental en el proceso de enseñanza-aprendizaje de la Teoría de Grafos?
2. ¿Qué aspectos deben tenerse en cuenta para realizar el análisis y diseño del módulo de Teoría de Grafos del Sistema Web interactivo y experimental para el proceso de enseñanza-aprendizaje de la Teoría de Grafos?
3. ¿Cómo implementar, a partir del análisis y diseño realizado, el módulo de Teoría de Grafos del Sistema Web interactivo y experimental?
4. ¿Qué resultados se obtendrán al validar el módulo de Teoría de Grafos del Sistema Web interactivo y experimental para el proceso de enseñanza-aprendizaje de la Teoría de Grafos?

Acorde a las preguntas científicas ya expuestas, se trazan los siguientes **objetivos específicos**:

1. Analizar los principales referentes teóricos que sustentan el desarrollo y la utilización del módulo de Teoría de Grafos del Sistema Web interactivo y experimental en el proceso de enseñanza-aprendizaje de la Teoría de Grafos.
2. Diseñar el módulo de Teoría de Grafos del Sistema Web interactivo y experimental.
3. Implementar el módulo de Teoría de Grafos del Sistema Web interactivo y experimental.
4. Validar el módulo de Teoría de Grafos del Sistema Web interactivo y experimental para el proceso de enseñanza-aprendizaje de la Teoría de Grafos.

Para el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- **Analítico-Sintético:** se empleó para examinar la bibliografía correspondiente a los contenidos de Matemática Discreta, con el objetivo de utilizar la información para la realización del módulo de Teoría de Grafos.
- **Análisis documental:** para la revisión bibliográfica, la revisión de las fuentes primarias de investigación, el estudio de documentos normativos, como el programa de la asignatura.
- **Modelación:** para la modelación de las funcionalidades del módulo. Permitió la abstracción de la realidad al modelo informático y de ahí buscar la manera de implementarlo, concretando las características de cada uno de los contenidos seleccionados.

Métodos Empíricos:

- **Observación:** se utilizó para constatar las deficiencias existentes en herramientas matemáticas utilizadas en la UCI, así como investigar sobre el funcionamiento, ventajas y desventajas de otras aplicaciones educativas similares.
- **Entrevista:** se utilizó con el objetivo de conocer acerca de la contribución del módulo desarrollado en el proceso de enseñanza-aprendizaje de la Teoría de Grafo.

El presente documento está estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica: este capítulo contiene la base teórica para entender el problema planteado, se realiza una valoración de la importancia de la Teoría de Grafos en la MD para la formación de futuros profesionales. Se argumenta el uso de las TIC en la educación caracterizando los Objetos de Aprendizaje (OA) como recursos de apoyo al Proceso de Enseñanza y Aprendizaje (PEA) de la Matemática Discreta. Finalmente se caracterizan las herramientas, lenguajes y metodologías estudiadas para la implementación.

Capítulo 2: Propuesta de solución: en este capítulo se presentan las fases de planificación y diseño definidas por la metodología Proceso Unificado Ágil (AUP, por sus siglas en inglés) variación UCI. Se realiza la propuesta de solución y se planifica el proceso de desarrollo de *software*. Se definen, entre otros, las historias de usuarios, los requisitos funcionales y no funcionales y la arquitectura del sistema.

Capítulo 3: Implementación y prueba de la propuesta de solución. Se definen los estándares de codificación a utilizar y se realiza el diagrama de componente y de despliegue de la solución. Además, se realizan pruebas para verificar la calidad del módulo de Teoría de Grafos, se muestran los resultados y las interfaces principales.

CAPÍTULO 1. Fundamentos teóricos de la investigación

1.1. Introducción.

En este capítulo se establecen conceptos fundamentales para la comprensión de la investigación. Se realiza un análisis del estado del arte de las herramientas educativas que presentan características similares al módulo que será desarrollado. Se realiza una valoración de la importancia de la Teoría de Grafos en la Matemática Discreta en la formación de los futuros profesionales de las TIC, se argumenta sobre el uso de las TIC en el proceso de enseñanza-aprendizaje, y se caracterizan los OA como recursos de apoyo al Proceso de Enseñanza y Aprendizaje de la Matemática Discreta. Finalmente se caracterizan las herramientas, lenguajes y metodologías consideradas para dar solución al problema de investigación.

1.2. Conceptos asociados a la investigación.

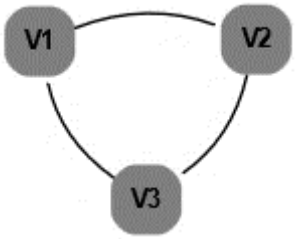
Grafos.

Un grafo es una pareja de conjuntos G = (V, E), donde V es el conjunto de vértices, y E es el conjunto de aristas definido como conjunto de pares de la forma (u, v) (Hinz, 2012)**.**

En el siguiente ejemplo se muestran las diferentes formas de representar un grafo:

$$V = \{v_1, v_2, v_3\}$$

$$E = \{v_1v_2, v_2v_3, v_1v_2\}$$



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Ilustración 1: Ejemplo de definición de un grafo (Elaboración propia).

Camino de Euler.

Dado G = (V, E) no orientado y conexo, sin vértices aislados. Un camino de Euler es una trayectoria que contiene todas las aristas de G y recorre cada arista exactamente una vez (Zumalacárregui, 2014).

Ciclo de Euler.

Sea G = (V, E) un grafo sin vértices aislados. Un ciclo de Euler es un camino que contiene todas las aristas de G, empieza y termina en el mismo vértice y recorre cada arista exactamente una vez (Herrera, 2015).

Camino de Hamilton.

Un camino hamiltoniano, en el campo matemático de la Teoría de Grafos, es un camino donde una sucesión de aristas adyacentes visita todos los vértices del grafo una sola vez (Lluch, 2013).

Ciclo de Hamilton.

Un ciclo de Hamilton es un camino hamiltoniano donde el primer vértice coincide con el último (Cortés, Bru, y Villalobos, 2016).

Árbol.

Un árbol es un grafo conexo y sin ciclos donde todo par de vértices está unido por un solo camino (Robledo, Osorio, y Lopez, 2015).

Además, son relevantes los conceptos de componente conexa, vértice de corte, arista de corte, grafo: bipartido, simple, completo, rueda, circular y conexo (Galvín y Javier, 2016).

En la siguiente tabla se muestra un resumen de los principales conceptos estudiados relacionados con la Teoría de Grafos, los cuales pueden ser encontrados en (Rosen, 2012).

Tabla 1: Conceptos relacionados con la Teoría de Grafos (Elaboración propia).

Conceptos relacionados con la Teoría de Grafos		
Grado del vértice	Vértice de corte	Componentes conexas
Arista de corte	Grafo regular	Grafo completo
Grafo rueda	Grafo circular	Grafo bipartido
Grafo bipartido completo	Grafo simple	Grafo conexo
Árbol	Cadena de Euler	Cadena cerrada de Euler
Grafo de Euler	Camino de Hamilton	Ciclo de Hamilton
Grafo de Hamilton	Camino	Ciclo

Interactividad: es la relación de participación entre los usuarios y los sistemas informáticos, es un proceso de comunicación entre humanos y computadoras; Rost (2014) se refiere a ella como la capacidad de las computadoras por responder a los requerimientos de los usuarios.

Software experimental: conceptualizado por Método Experimental (2012) como un programa basado en la observación y manipulación sistemática de la realidad para evolucionar el conocimiento que se tiene sobre un fenómeno concreto; implica la observación, manipulación, registro de variables (dependientes, independientes, intervinientes, etc.) que afectan un objeto de estudio.

Objetos de aprendizaje.

Según Churchill (2007) un objeto de aprendizaje es un conjunto de recursos digitales, auto contenible y reutilizable, con un propósito educativo y constituido por al menos tres componentes internos: contenidos, actividades de aprendizaje y elementos de contextualización.

- **Contenidos:** se refiere a los tipos de conocimiento y sus múltiples formas de representarlos, pueden ser: definiciones, explicaciones, artículos, videos, entrevistas, lecturas, opiniones, incluyendo enlaces a otros objetos, fuentes, referencias, entre otros.
- **Actividades de aprendizaje:** se refiere a las actividades que guían al estudiante para alcanzar los objetivos propuestos.
- **Elementos de contextualización:** permiten reutilizar el objeto en otros escenarios, como por ejemplo los textos de introducción, el tipo de licenciamiento y los créditos del objeto.

Gamificación.

La gamificación es el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos. Se trata de una nueva y poderosa estrategia para influir y motivar a grupos de personas (Espinosa, 2016)**.**

También puede ser definida como una técnica de aprendizaje que traslada la mecánica de los juegos al ámbito educativo-profesional con el fin de conseguir mejores resultados, ya sea para absorber mejor algunos conocimientos, mejorar alguna habilidad, o bien recompensar acciones concretas, entre otros muchos objetivos (Arteta, 2015).

Este tipo de aprendizaje gana terreno en las metodologías de formación debido a su carácter lúdico, que facilita la interiorización de conocimientos de una forma más divertida, generando una experiencia positiva en el usuario. El modelo de juego realmente funciona porque consigue motivar a los alumnos, desarrollando un mayor compromiso de las personas, e incentivando el ánimo de superación. Se utilizan una serie de técnicas mecánicas y dinámicas extrapoladas de los juegos (Díaz, 2015).

La técnica mecánica definida por González y Carreño (2015) es la forma de recompensar al usuario en función de los objetivos alcanzados. Algunas de las técnicas mecánicas más utilizadas son las siguientes:

- **Acumulación de puntos:** se asigna un valor cuantitativo a determinadas acciones y se va acumulando a medida que se realizan.

- **Escalado de niveles:** se define una serie de niveles que el usuario debe ir superando para llegar al siguiente.
- **Obtención de premios:** a medida que se consigan diferentes objetivos se van entregando premios a modo de colección.
- **Regalos:** bienes que se dan al jugador o jugadores de forma gratuita al conseguir un objetivo.
- **Clasificaciones:** clasificar a los usuarios en función de puntos u objetos logrados, destacando los mejores en una lista o *ranking*.
- **Desafíos:** competiciones entre los usuarios, el mejor obtiene los puntos o el premio.

Las técnicas dinámicas presentadas por Sánchez y Rivero (2015) hacen referencia a la motivación del propio usuario para jugar y seguir adelante en la consecución de sus objetivos. Algunas de las técnicas dinámicas más utilizadas son las siguientes:

- **Recompensas:** obtener un beneficio merecido.
- **Estatus:** establecerse en un nivel jerárquico social valorado.
- **Logro:** como superación o satisfacción personal.
- **Competición:** por el simple afán de competir e intentar ser mejor que los demás.

1.3. Análisis de sistemas homólogos.

MaGraDa.

El paquete de software MaGraDa (Grafos para Matemática Discreta) es una aplicación informática programada en lenguaje *JAVA*[2] y diseñada específicamente para trabajar con grafos, tanto dirigidos como no dirigidos y ponderados como no ponderados. Según la filosofía de MaGraDa puede trabajar con un número ilimitado de vértices, pero se estableció conveniente sólo permitir trabajar con grafos de hasta un máximo de 50 vértices (Palomino, Gomis, y Martínez, 2006).

Este software consta de dos pantallas de visualización:

- **Modo texto:** permite trabajar con los grafos de forma analítica.
- **Modo gráfico:** trabaja con los grafos de forma que pueden verse gráficamente.

Ambas pantallas de trabajo son prácticamente equivalentes en funcionalidad. En cada uno de ellos se muestran los resultados intentando maximizar la comprensión de los mismos por el usuario. Básicamente, las aplicaciones que ofrece MaGraDa se agrupan en tres partes:

[2] Lenguaje de programación de propósito general, concurrente, orientado a objetos.

- **Manejo de grafos (Grafo):** permite crear grafos nuevos o abrir grafos ya creados desde un fichero, modificarlos, borrarlos de memoria, seleccionarlos o guardarlos en un fichero para su tratamiento posterior (Palomino, Gomis, & Martínez, 2006).
- **Cálculos Básicos:** existe una serie de características o propiedades básicas de los grafos que se pueden averiguar fácilmente con esta serie de métodos, tales como: grado de un vértice, matriz de adyacencia o pesos, ver aristas (o arcos) que pueda tener el grafo. También, en el caso de que el grafo sea dirigido, MaGraDa ofrece la utilidad de obtener su correspondiente grafo no dirigido asociado. Para grafos no dirigidos obtiene un árbol generador de los muchos que pueda tener. Permite estudiar, también desde este menú, si dos grafos son isomorfos, ver qué vértices alcanzan a otros, así como qué vértices son alcanzados por otros. Indica además si el grafo es simple, cíclico, completo o conexo. Otra aplicación de gran interés es el cálculo de componentes conexas (Palomino, Gomis, y Martínez, 2006).
- **Algoritmos:** dispone de algoritmos muy conocidos en el mundo de los grafos, tales como: *Warshall*, *Fleury*, Caminos más cortos en grafos acíclicos, *PERT*, *Dijkstra*, *Floyd-Warshall*, *Kruskal* y *Prim*. Destacar que MaGraDa los aplica sobre los grafos en curso, de manera que el usuario pueda ver los resultados intermedios para así entender mejor el funcionamiento del correspondiente algoritmo (Palomino, Gomis, y Martínez, 2006).

ProRouting.

Prorouting (*PROximity Graphs and routing*) es una herramienta diseñada para el estudio de grafos geométricos, en particular de los grafos de proximidad, las estrategias de ruteo en ellos y sus propiedades (Universidad Politécnica de Madrid (UPM), 2015).

La herramienta se empezó a desarrollar en 2004 y desde entonces se le han añadido numerosas mejoras y aumento de funciones. En este momento está disponible la versión 2.

A continuación, se describen algunas de las características de la aplicación:

Grafos de proximidad y estructuras geométricas en *Prorouting*

- Grafo del vecino más cercano. Grafo de los *k*-vecinos más próximos. Par más próximo.
- Cierre convexo.
- Árbol generador de peso mínimo. Árbol generador de peso mínimo k-local.
- Grafo del disco unidad.
- Grafo de Vecindad Relativa.

- Grafo de Gabriel.
- Triangulación incremental. Triangulación voraz.
- Triangulación de Delaunay. Diagrama de Voronoi.
- Beta-esqueletos.
- Grafo de proximidad por semiplanos.
- Familia de grafos *theta-lambda*.
- *Theta* grafo. *Theta* grafo dirigido. *Theta* grafo ordenado. "*Sink spanner*".
- Grafo de Yao. Grafo de Yao dirigido. Grafo Yao-Yao. Grafo de Yao simétrico.

Estrategias de ruteo en *Prorouting*

- Ruteo voraz.
- Ruteo por brújula. Ruteo aleatorizado por brújula.
- Ruteo híbrido voraz-brújula.
- Ruteo Voronoi.
- Ruteo por caras (variantes 1 y 2). Ruteo voraz-cara-voraz.
- Ruteo ad-hoc para grafos *theta-lambda*.
- Estudio de la calidad de todas las estrategias de ruteo.

Construcción de caminos y *spanners* en *Prorouting*

- Caminos de longitud mínima en un grafo (algoritmo de *Dijkstra*).
- Algoritmo *PathGreedy.*
- Algoritmo *Sparse Spanner.*

AlGraf.

Algraf (Algoritmos en Grafos) es una herramienta diseñada en *Visual Basic* para el estudio de los algoritmos sobre grafos. El objetivo de la aplicación es fundamentalmente didáctico, para ayudar a la mejor comprensión de algunos conceptos y para visualizar de forma animada algoritmos sobre grafos (Universidad Politécnica de Madrid (UPM), 2015).

El programa responde a numerosas cuestiones sobre un grafo o dígrafo (con o sin pesos en las aristas):

- Sucesión de grados, matriz de adyacencia.
- Conectividad, vértices de corte, aristas puente, componentes conexas y bloques.
- Operaciones sobre grafos: grafo complementario y grafo de aristas *(line graph*).

- Árboles: algoritmos de búsqueda, algoritmos de construcción del árbol generador de peso mínimo (*Prim, Kruskal y Boruvka*) y código de *Prüfer* de un árbol etiquetado.
- Caminos en un grafo o dígrafo sin pesos o con pesos (algoritmo de *Dijkstra*).
- Recorridos eulerianos: existencia, construcción con los algoritmos de *Hierholzer, Fleury y Tucker.*
- Problema del cartero.
- Coloración: algoritmos secuenciales, variantes y algoritmo de *Brelaz.*

Wolfram Research Mathematica.

Mathematica es un programa utilizado en áreas científicas, de ingeniería, matemáticas y computacionales. Originalmente fue concebido por Stephen Wolfram, quien continúa siendo el líder del grupo de matemáticos y programadores que desarrollan el producto en *Wolfram Research*, compañía ubicada en *Champaign*, *Illinois*. Comúnmente considerado como un sistema de álgebra computacional, *Mathematica* es también un poderoso lenguaje de programación de propósito general (Intercambios Virtuales, 2017).

Este software presenta las siguientes características:

- Bibliotecas de funciones elementales y especiales para matemáticas.
- Herramientas de visualización de datos en 2D y 3D.
- Matrices y manipulación de datos, así como soporte de matrices tipo "*sparse*".
- Capacidad de solucionar sistemas de ecuaciones, ya sea ordinarias, parciales o diferenciales, así como relaciones de recurrencia y algebraicas en general.
- Herramientas numéricas y simbólicas para cálculo de variable continua o discreta.
- Estadística multivariable.
- Lenguaje de programación que soporta programación funcional.
- Un kit de herramientas para añadir interfaces de usuario para cálculos y aplicaciones.
- Herramientas para procesamiento de imágenes.
- Herramientas de análisis y visualización.
- Minería de datos, como análisis de *clusters*, alineamiento de secuencias, y "*pattern matching*".

Grafos.

Este software pretende ser de utilidad para la docencia y el aprendizaje de la Teoría de Grafos, y otras disciplinas relacionadas como la ingeniería de organización industrial, la logística y el transporte, investigación operativa y el diseño de redes. Grafos se puede usar perfectamente para el modelado y

resolución de problemas reales de cierto tamaño y complejidad (Rodríguez, 2006).

El software está orientado hacia los siguientes objetivos:

- Desarrollar un interfaz para la construcción y edición de grafos en modo tabular o gráfico, y que permita la incorporación modular de multitud de funciones.
- Desarrollar una estructura de clases y librerías *.dll* con un conjunto de algoritmos de resolución de diferentes problemas de Teoría de Grafos.

Conclusiones del análisis de las soluciones existentes.

Luego de realizado un análisis de los sistemas informáticos que se utilizan en la enseñanza y el aprendizaje de la Teoría de Grafos se concluye que las aplicaciones estudiadas presentan dificultades en cuanto a la capacidad de retroalimentar al profesor acerca de las tareas realizadas por sus estudiantes, todas son aplicaciones de escritorio, por lo que es necesario tenerlas instaladas para su uso y solo algunos permiten la experimentación con la Teoría de Grafos, no obstante se tendrán en cuenta para el desarrollo de la propuesta de solución.

De ellos se puede obtener experiencia acerca de los principales algoritmos que se pueden incluir en el sistema y de la forma en la que los usuarios introducen los grafos de manera gráfica, lo cual hace menos engorrosa la entrada de datos.

Además, a pesar de que todos los sistemas estudiados están diseñados con fines educativos y permiten el trabajo ya sea con grafos dirigidos y ponderados, poseen visualización en dos y tres dimensiones y además realizan muchos de los algoritmos más utilizados en la Teoría de Grafos como; *Prim y Kruskal,* ninguno permite una interacción constante entre los estudiantes y el profesor, no les permite a los estudiantes generar sus propios ejercicios para desarrollar sus habilidades y tampoco tienen estrategias de gamificación para aumentar la motivación de los estudiantes.

El sistema *Mathematica* de la compañía *Wolfram* es uno de los más usados en la educación para el proceso de enseñanza-aprendizaje, pero además de no cumplir con las principales exigencias de un sistema interactivo y experimental, también necesita de una licencia de pago por cada computadora en la que se instale la aplicación.

La siguiente tabla muestra un resumen de algunas de las características antes descritas.

Tabla 2: Resumen comparativo de soluciones existentes (Elaboración propia).

Herramienta	Experimentación	Web	Gamificación
MaGraDa	Limitado	NO	NO
Prorouting	SÍ	NO	NO
AlGraf	Limitado	NO	NO
Mathematica	SÍ	NO	NO
Grafos	Limitado	NO	NO

1.4. Ambiente de desarrollo a emplear en la propuesta de solución.

Para el desarrollo del módulo Teoría de Grafos del sistema web interactivo y experimental de apoyo al proceso de enseñanza-aprendizaje de la Matemática Discreta, es necesario realizar un estudio de las herramientas, lenguajes y tecnologías utilizadas en el desarrollo de la plataforma en la cual se integrará el módulo.

1.4.1. Metodologías de desarrollo de software.

En la Ingeniería de Software, una metodología de desarrollo de software es la encargada de la separación de este proceso en distintas fases o etapas, que contienen actividades enfocadas a una mejor planificación y administración del mismo. Puede incluir la definición previa de una serie de artefactos que son creados y completados con el equipo que desarrolla o mantiene la aplicación (CMS, 2008).

Existen numerosas propuestas metodológicas que inciden en diferentes dimensiones del mismo. Hay propuestas más tradicionales que se centran especialmente en el control, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben generar, así como las herramientas a utilizar. Otras se enfocan en el factor humano o el producto de software, esta filosofía de las metodologías ágiles da mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas (Canós, Letelier, y Penadés, 2003).

Para el desarrollo de la solución propuesta se seleccionó Proceso Unificado Ágil (AUP, por sus siglas en inglés), versión UCI, debido a que logra estandarizar el proceso de desarrollo de software, dando

cumplimiento a las buenas prácticas que define CMMI-DEV[3] v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos.

1.4.2. Modelado de software.

El modelado de software es una técnica para tratar con la complejidad inherente a estos sistemas. El uso de modelos ayuda al ingeniero de software a "visualizar" el sistema a construir. Además, los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. Por último, las herramientas de modelado y las de Ingeniería de Software Automatizada pueden ayudar a verificar la corrección del modelo (Academic, 2012). **Lenguaje Unificado de Modelación (*UML*).**

UML en su versión 2.1, 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 de software. UML ofrece un estándar para describir un plano o modelo del sistema, incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Cortés, Hernández, y Cabrera, 2017).

Herramienta para el modelado.

Para modelar el módulo se hará uso de una herramienta de Ingeniería de Software Asistida por Computadoras (*CASE*) por sus siglas en inglés. Las herramientas *CASE* son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el diseño de proyectos, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras (Sommerville, Technology y Engineering, 2005).

Visual Paradigm en su versión 8.0, se selecciona como herramienta *CASE* de modelado profesional, que utiliza *UML* para la completa representación de las etapas por las que transita un producto de software. Este permite la realización de una amplia gama de diagramas como: casos de uso, de actividades, de despliegue, entre otros, así como la generación de código fuente desde los mismos y la documentación asociada al proceso que esté siendo modelado (Visual Paradigm, 2013).

[3] Modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

1.4.3. Lenguaje de programación.

***Python*.**

Python en su versión 3.4.1, es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su elegante sintaxis y tipos de datos dinámicos junto con su naturaleza interpretada, hacen de este un lenguaje ideal para *scripting* y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas (Sevilla, Fernández, y Díaz., 2016).

JavaScript.

JavaScript es un lenguaje de programación interpretado, dialecto del estándar *ECMAScript*[4]. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente en su forma del lado del cliente (*client-side*), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas (Park, Stefănescu y Roşu, 2015).

Librería *jQuery*.

JQuery, es una librería multiplataforma de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML[5], manipular el árbol DOM[6], manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX[7] a páginas web. Es además un software libre y de código abierto, permitiendo su uso en proyectos tanto libres como privados. JQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta, se logran grandes resultados en menos tiempo y espacio (Gutiérrez, 2009).

Librería GoJS

GoJS, es una librería de JavaScript con múltiples funciones para implementar diagramas interactivos personalizados y visualizaciones complejas en navegadores y plataformas web modernos. Facilita la construcción de diagramas con nodos, enlaces y grupos complejos, con plantillas y diseños personalizables. Ofrece, además, muchas características avanzadas para la interactividad del usuario, tales como arrastrar

[4] Especificación de lenguaje de programación publicada por *ECMA International***,** organización internacional basada en membresías de estándares para la comunicación y la información.
[5] Del inglés HyperText Markup Language (Lenguaje de marcas de hipertexto)
[6] Del inglés Document Object Model (Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos).
[7] Del inglés, acrónimo de JavaScript asíncrono y XML.

y soltar, copiar y pegar, edición de texto en el lugar, información sobre herramientas, menús contextuales, diseños automáticos, plantillas, vinculación y modelos de datos, administración de estados transaccionales y deshacer, paletas, vistas generales, controladores de eventos, comandos y un sistema de herramientas extensible para operaciones personalizadas (GoJS, 2017).

GoJS es puro *JavaScript*, por lo que los usuarios obtienen interactividad sin necesidad de recorridos de ida y vuelta a servidores y sin complementos. Normalmente se ejecuta completamente en el navegador, mostrando a un elemento de lienzo HTML5 sin requisitos de servidor. No depende de librerías o *frameworks* JavaScript, por lo que debería funcionar con cualquier framework HTML o JavaScript o sin estos (GoJS, 2017).

Lenguaje marcado de hipertexto (HTML 5).

Es un nuevo lenguaje de marcado de hipertexto para presentar y estructurar el contenido en internet. Es la quinta revisión y nueva versión del estándar HTML. HTML5 ofrece nuevas características que proporcionan no sólo un rico soporte multimedia (video y audio), sino que también mejoran el apoyo para la creación de aplicaciones web. Esta nueva tecnología puede proporcionar una solución rentable para implementar aplicaciones para apoyar diferentes dispositivos (por ejemplo, ordenador, tableta, teléfono inteligente personal) sin tener que construir aplicaciones diferentes para cada tipo de dispositivos (Gauchat, 2014).

Hojas de Estilo en Cascada (CSS 3).

Hojas de estilo en cascada (o *CSS*, siglas en inglés de *Cascading Stylesheets*) es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado. Es muy usado para establecer el diseño visual de las páginas web, e interfaces de usuario escritas en *HTML* o *XHTML*; el lenguaje puede ser aplicado a cualquier documento *XML*, incluyendo *XHTML*, *SVG*, *XUL*, *RSS*. También permite aplicar estilos no visuales, como las hojas de estilo auditivas (Sebastián, 2015).

1.4.4. Marco de Trabajo.

Se seleccionó *Django* como *framework* de desarrollo web del lado del servidor, pues, es construido sobre *Python*, fomenta el desarrollo ágil permitiendo la construcción de aplicaciones de alta calidad en un breve período de tiempo. Utiliza una modificación de la arquitectura Modelo – Vista – Controlador (MVC) llamada Modelo – Plantilla – Vista (*MTV*, por sus siglas en inglés). *Django* impulsa el desarrollo de código limpio y

promueve la utilización de buenas prácticas de desarrollo web como el principio *DRY*[8]. Entre algunos casos de éxito se encuentran: *Instagram*[9], *Pinterest*[10] y *The New York Times* (Condori, 2012).

1.4.5. Sistema Gestor de Base Datos.

PostgreSQL en su versión 9.4.1, es un Sistema Gestor de Bases de Datos (SGBD) objeto-relacional que se encuentra distribuido bajo la licencia de software libre BSD[11]. Es mantenido por la organización *PostgreSQL Global Development Team* y cuenta con una amplia comunidad de usuarios y programadores que colaboran activamente. Se centra en que el software sea robusto, de calidad, fácil de mantener y se destaca por su estabilidad, potencia, robustez y facilidad de administración. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios, accediendo paralelamente al sistema (Martínez, 2010).

1.4.6. Entorno Integrado de Desarrollo.

PyCharm en su versión 2016.3.2, es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación *Python*. Provee funcionalidades que permiten una experiencia única y aumentan en gran medida la productividad:

- Completamiento de código de manera inteligente y señalamiento de errores con reparación de los mismos de forma automatizada.
- Integración con marcos de trabajo de desarrollo web modernos como *Django*.
- Depurador integrado y herramientas de pruebas.
- Soporte para bases de datos e integración con sistemas de control de versiones.

En adición a *Python*, *PyCharm* soporta *JavaScript*, *CoffeeScript*, *TypeScript*, *HTML/CSS, Cython*, lenguajes de plantilla, *AngularJS*, *Node.js* y más (JetBrains, 2016).

1.4.7. Servidor Web.

Un servidor web es un programa que utiliza el protocolo de transferencia de hipertexto, *HTTP* (*Hypertext Transfer Protocol),* para servir los archivos que forman páginas web a los usuarios, en respuesta a sus solicitudes, que son reenviados por los clientes *HTTP* de sus computadoras (Soni, 2016).

[8] Principio orientado a reducir la repetición de información de todo tipo, especialmente útil en arquitecturas de múltiples niveles.
[9] Red social y aplicación para subir fotos y videos. Disponible en: https://www.instagram.com/
[10] Plataforma para compartir imágenes. Disponible en https://es.pinterest.com/
[11] Licencia de software libre permisiva como la licencia de *OpenSSL*.

Apache HTTP Server.

Apache es una aplicación gratuita que convierte un ordenador en un servidor web. Es de código abierto, flexible, rápido y eficiente. *Apache* permite negociar protocolos *HTTP* entre una máquina que haría de servidor web y los otros ordenadores que deseen ver un determinado sitio web. Es multiplataforma y gracias a su popularidad es fácil encontrar documentación para profundizar en las funcionalidades que ofrece. Se caracteriza por su gran escalabilidad, seguridad y rendimiento. Tiene soporte para varios lenguajes como *Perl*, *Python* y *PHP*, lo que permite desarrollar aplicaciones web de gran calidad (Apache Software Foundation, 2011).

1.5. Conclusiones parciales del capítulo.

El estudio de los conceptos relacionados con el dominio de la investigación permitió un mejor entendimiento del negocio. Mediante el análisis de los sistemas homólogos además de obtener experiencia acerca de los principales algoritmos que se pueden incluir en el módulo, se observaron algunas carencias que poseen los mismos en cuanto al trabajo de la Teoría de Grafos, por tal motivo se ratifica la necesidad de una nueva herramienta y la selección de las tecnologías que permitan dar comienzo al desarrollo de software. Las herramientas seleccionadas facilitan el desarrollo en cuanto al modelado y la implementación de la aplicación, de forma que se logre una correcta integración con el sistema interactivo y experimental para el proceso enseñanza-aprendizaje de la Matemática Discreta, guiada por la metodología de desarrollo de software *AUP* versión UCI.

CAPÍTULO 2. Análisis y diseño de la herramienta

2.1 Introducción.

En este capítulo se presentan las fases de planificación y diseño definidas por la metodología *AUP* variación UCI escenario cuatro (4). Se realiza la propuesta de solución y se planifica el proceso de desarrollo de software, identificando los requerimientos funcionales y no funcionales de la propuesta de solución, patrones de diseño y estilo arquitectónico.

2.2 Características de la propuesta de solución.

Una vez analizada la situación problemática de la investigación, la solución propuesta consiste en un módulo para el entrenamiento de habilidades en el tema Teoría de Grafos de la asignatura Matemática Discreta, que no se encuentra actualmente en el sistema interactivo experimental.

Estructura y funcionamiento del módulo.

El módulo de Teoría de Grafos contará de tres (3) unidades para lograr una buena organización de los procesos de enseñanza-aprendizaje, las cuales se definen a continuación.

Unidad de Información: el estudiante tendrá a su disposición toda la bibliografía del tema, en formatos de: texto, imágenes, videos, audios, entre otros, permitiendo descargar los ficheros o utilizarlos dentro de la aplicación.

Dentro de la bibliografía del tema debe encontrarse dinámicamente el libro “Matemáticas Discretas para Ingenieros en Ciencias Informáticas” del MSc. Alién García Hernández.

Unidad Didáctica: consiste en la experimentación, donde se le permita al estudiante crear grafos no dirigidos y sin ponderación a través de un panel de dibujo, de igual manera, el sistema debe permitir importar y exportar grafos en formato “. *json”* o desde el portapapeles de la aplicación. Una vez definidos los grafos, el estudiante puede comenzar la experimentación, esta consiste en una interacción directa con cada uno de los componentes del grafo, permitiendo ver los resultados que se generan al seleccionar las diferentes funcionalidades definidas en dicha unidad, permitiendo a los estudiantes generar sus propios ejercicios para su auto-evaluación y auto-preparación en la asignatura Matemática Discreta, donde el estudiante genera los grafos permitiéndole experimentar con las siguientes operaciones: determinar el grado de un vértice, si una secuencia de vértices es un camino y/o cadena cerrada de Euler, determinar si un vértice o una arista es de corte, si el grafo es simple, circular, rueda, regular, bipartido, completo, bipartido completo, conexo, contiene una cadena de Euler, cadena cerrada de Euler, un ciclo de Hamilton o un camino de Hamilton.

Unidad de Evaluación y Retroalimentación: contiene cuatro variables las cuales son: evaluaciones, autoevaluaciones, competencias en línea y estrategias de gamificación.

Para esta unidad se le debe permitir al profesor crear las evaluaciones con una cantidad de ejercicios definidos por él, las cuales el estudiante podrá resolver antes de una fecha de caducidad especificada, estas pueden estar protegidas con contraseña y tener acceso restringido por subredes. Cuando el estudiante resuelve la evaluación, el sistema debe ser capaz de emitir una nota parcial, la cual puede ser modificada por el profesor ya que las evaluaciones pueden incluir justificación por parte del estudiante o que la respuesta sea el envío de un archivo de imagen, el sistema asigna la nota de acuerdo al porciento de preguntas correctas que tenga el estudiante en la evaluación de la siguiente manera:

- Si el porciento de respuestas correctas es menor que sesenta (60), la nota será de dos (2) puntos.
- Si el porciento está entre sesenta (60) y ochenta (80), la nota será de tres (3) puntos.
- Si el porciento está entre ochenta (80) y noventa (90), la nota será de cuatro (4) puntos.
- Si el porciento está entre noventa (90) y cien (100), la nota será de cinco (5) puntos.

Las autoevaluaciones deben funcionar de igual manera, solo que la nota no interfiere en el desempeño del estudiante, no tiene límites de intentos, están disponible para toda la red y nunca se protegen con contraseña.

Las competencias se evalúan igual que las evaluaciones, pueden estar protegidas con contraseña, pero no interfieren en el índice académico que obtiene el estudiante en la aplicación. Las competencias solamente aportan puntos a los estudiantes según su desempeño, a través de una bonificación que el profesor le asigna a la evaluación competitiva, los puntos de bonificación se asignan de la siguiente manera:

- Si la nota es de cinco (5) puntos el estudiante obtiene todos los puntos de la bonificación.
- Si la nota es de cuatro (4) puntos el estudiante pierde el diez (10) por ciento de los puntos.
- Si la nota es de tres (3) puntos el estudiante pierde el veinte (20) por ciento de los puntos.
- Si la nota es de dos (2) puntos el estudiante pierde la bonificación completa y se le descuenta de su acumulado de puntos en las competencias.

Estrategia de Gamificación utilizada.

Para la implementación de la estrategia de gamificación utilizada en el módulo de Teoría de Grafos se tomó como guía la técnica mecánica definida por González y Carreño (2015) la cual consiste en recompensar al usuario en función de los objetivos alcanzados.

En función de las bonificaciones obtenidas se definieron una serie de parámetros para implementar técnicas de gamificación como fueron, la acumulación de puntos, obtenidos por el estudiante según su desempeño

en las competencias, donde los puntos de la bonificación son en dependencia de la nota, como se explicó anteriormente. La escalabilidad de niveles, es otra de las técnicas utilizadas donde en dependencia de los puntos alcanzados el sistema le asigna el nivel de principiante, aprendiz, aventajado, experto o súper-estrella.

Otra técnica utilizada fue la obtención de premios donde de igual manera en dependencia de los puntos acumulados por cada estudiante se le asignan premios virtuales como se observa en la siguiente ilustración.

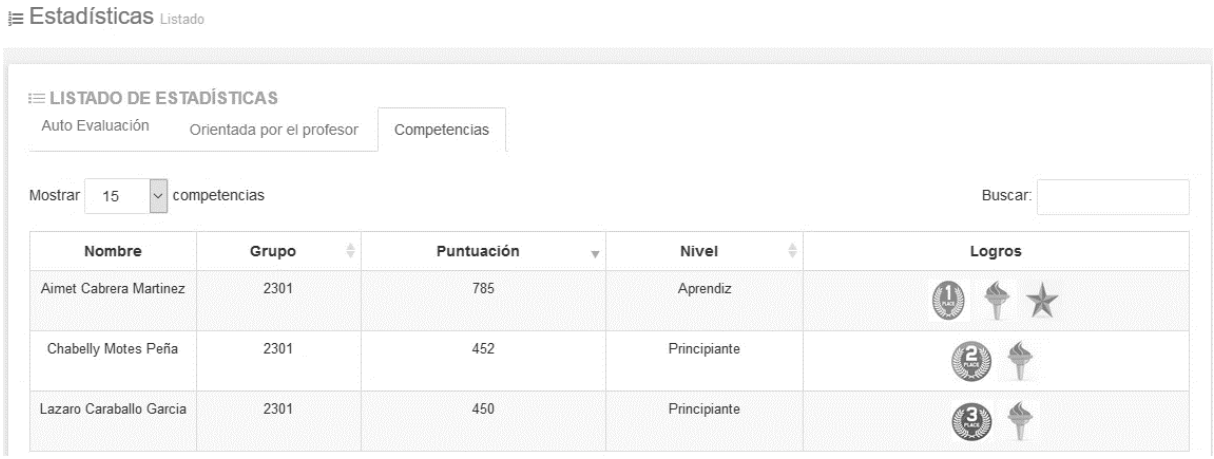


Ilustración 2: Página de Estadísticas, recurso de gamificación (Elaboración propia).

Finalmente, el sistema debe permitir establecer un *ranking* de los mejores resultados, el cual se establece por el total de puntos acumulados por cada estudiante.

2.3 Modelo conceptual.

Un modelo conceptual es una representación visual en forma de diagrama de las clases conceptuales u objetos del mundo real que son significativos en un dominio de interés; no se trata de un conjunto de diagramas que describen clases u objetos de software con responsabilidades (Larman, 2004).

Los modelos conceptuales pueden utilizarse para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Similares a los mapas mentales utilizados en el aprendizaje, el modelo conceptual es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir (Losavio y Esteves, 2015).

Componen el modelo conceptual definido para la presente investigación:

UCI: Universidad de las Ciencias Informáticas, centro de altos estudios donde se pondrá en práctica el

módulo de Teoría de Grafos.

Matemáticas Discretas: asignatura para la cual se desarrolla el módulo de Teoría de Grafos.

Estudiante: usuario que usará el módulo de Teoría de Grafos, para su estudio individual y realizar tareas y evaluaciones orientadas por los profesores.

Profesor: usuario que se encarga de brindar la información necesaria para el estudio individual de los estudiantes y orientar los diferentes tipos de evaluación.

Tema: contiene todos los temas de la asignatura Matemáticas Discretas, donde se encuentra la Teoría de Grafos.

Experimentación: unidad didáctica que contiene una serie de propiedades de los grafos, con las cuales se puede experimentar para observar el cumplimiento de cada uno de ellas, por un grafo dibujado por el propio estudiante.

Evaluación: unidad de evaluación y retroalimentación que contiene todas las evaluaciones definidas por los profesores para los estudiantes.

Contenido: unidad de Información, donde se encuentra toda la documentación del tema Teoría de Grafos.

Competencia: unidad de evaluación y retroalimentación donde se encuentran las evaluaciones competitivas.

Teoría de Grafos: tema de la asignatura Matemáticas Discretas.

Gamificación: estrategia que se usa en la unidad de evaluación y retroalimentación para apoyar la motivación de los estudiantes al estudio de las Matemáticas Discretas.

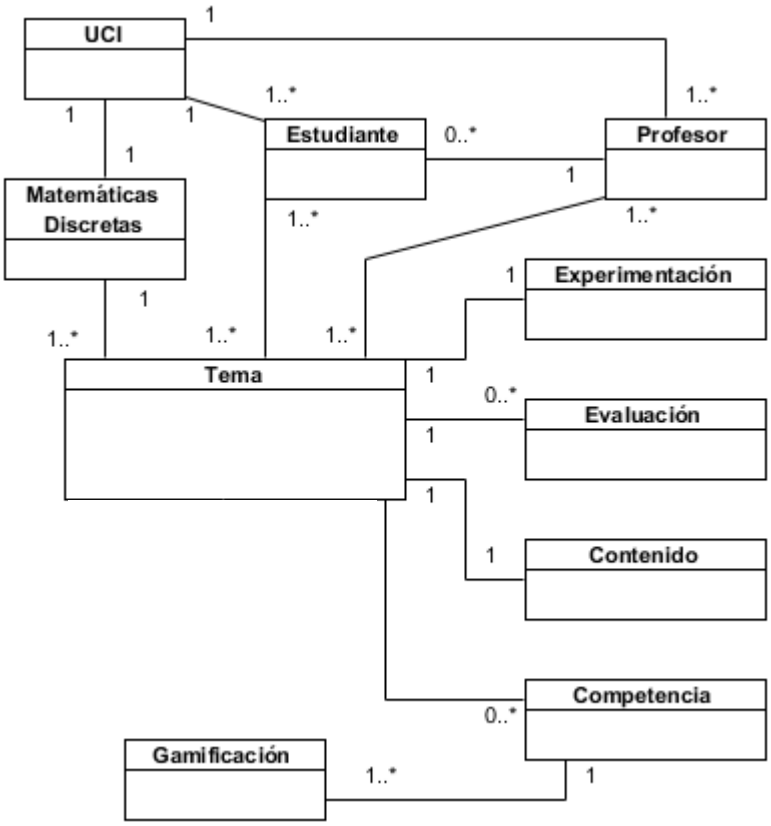


Ilustración 3: Modelo de dominio (Elaboración propia).

2.4 Especificación de requisitos.

Según Martínez e Imelda (2015), un requisito es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste. En el otro extremo, es una definición detallada y formal de una función del sistema.

Requisitos funcionales.

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone que un sistema debe cumplir (Sommerville, 2006).

Tabla 3: Requisitos Funcionales (Elaboración propia).

Funcionalidades	Prioridad
RF1: Experimentar con el tema “Teoría de Grafos”.	Alta
RF2: Exponer el contenido teórico del tema “Teoría de Grafos”.	Baja
RF3: Adicionar evaluaciones del tema “Teoría de Grafos”.	Alta
RF4: Deshabilitar evaluaciones del tema “Teoría de Grafos”.	Alta
RF5: Habilitar evaluaciones del tema “Teoría de Grafos”.	Alta
RF6: Listar evaluaciones del tema “Teoría de Grafos”.	Alta
RF7: Realizar evaluaciones orientadas por el profesor del tema “Teoría de grafos”.	Alta
RF8: Realizar autoevaluaciones del tema “Teoría de Grafos”.	Alta
RF9: Realizar competencias del tema “Teoría de Grafos”	Alta
RF10: Adicionar ejercicios del tema “Teoría de Grafos”.	Alta
RF11: Eliminar ejercicios del tema “Teoría de Grafos”.	Alta
RF12: Mostrar ayuda en las experimentaciones.	Baja
RF13: Mostrar *ranking* de usuarios según su desempeño en las competencias.	Alta
RF14: Asignar puntos a los usuarios según sus resultados en los ejercicios competitivos.	Alta
RF15: Asignar niveles para la superación de los usuarios en los ejercicios de competencias.	Alta
RF16: Asignar premios a los usuarios según sus resultados en los ejercicios competitivos.	Alta
RF17: Representar grafos gráficamente.	Alta
RF18: Exportar grafos en los formatos *.json* y *.png*.	Media
RF19: Importar grafo en los formatos *.json*.	Media
RF20: Guardar grafo en el portapapeles de la aplicación.	Media
RF21: Cargar grafo desde el portapapeles de la aplicación.	Media

Requisitos no funcionales.

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales (Ecured, 2017)**.**

Estándares:

RNF1: todas las clases y funciones deben quedar bien documentadas.

RNF2: implementar códigos robustos donde se comprueben todos los datos, evitando así colisiones inesperadas.

Rendimiento:

RNF3: la velocidad de procesamiento de la información y los tiempos de respuestas deben ser menores a los 5000 milisegundos.

RNF4: ante los errores que puedan ocasionarse en el sistema no se deben mostrar detalles de información que puedan comprometer su seguridad e integridad de los datos.

Interfaz:

RNF5: se debe mantener la misma distribución y diseño de los elementos del resto del sistema.

Usabilidad:

RNF6: se debe mostrar al usuario todos los detalles sobre los errores ocurridos en su petición, sin comprometer la seguridad del sistema.

Seguridad:

RNF7: se debe restringir el acceso a los recursos en dependencia de los permisos del usuario autenticado.

2.5 Historias de usuario.

En la metodología de desarrollo de software *AUP* variación UCI los requisitos que debe cumplir el software son especificados por los clientes en las denominadas historias de usuario. Estas historias son descompuestas en tareas de programación y asignadas a los programadores. A continuación, se describen las historias de usuario del sistema de acuerdo a la siguiente plantilla:

Tabla 4: Plantilla para las historias de usuario (Elaboración propia).

Historia de Usuario	
Número: *Número de la Historia de Usuario (HU), incremental en el tiempo.*	**Usuario:** *El usuario del sistema que utiliza o protagoniza la historia.*
Nombre de historia: *El nombre de la HU, sirve para identificarla fácilmente entre los desarrolladores y los clientes.*	
Prioridad en negocio: *Qué tan importante es para el cliente (Alta, Media o Baja).*	**Riesgo en desarrollo:** *Qué tan difícil es para el desarrollador (Alto, Medio o Bajo)*
Programador Responsable: *Persona que se encarga de la implementación de la funcionalidad.*	
Descripción: *La descripción de la historia, detallando las operaciones del usuario y opcionalmente las respuestas del sistema.*	

Para el diseño de la propuesta de solución fueron generadas un total de veintiuna (21) Historias de Usuarios (HU), a continuación, solo se muestran tres (3) de ellas correspondientes a los requisitos de mayor prioridad.

Tabla 5: Historia de usuario Experimentar con el tema "Teoría de Grafos" (Elaboración propia).

Historia de Usuario	
Número: HU_1	**Usuario:** Estudiante
Nombre de historia: Experimentar con el tema "Teoría de Grafos".	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Osmel Mojena Dubet.	
Descripción: El usuario dibuja un grafo en el panel de dibujo o importa un grafo desde un fichero *.json* o desde el portapapeles de la aplicación, luego tiene a su disposición una serie de conceptos y propiedades de los grafos las cuales puede comprobar si el grafo dibujado cumple con ellas, en caso del estudiante no recordar en que consiste cada una de esas propiedades, puede consultar la ayuda que brinda la aplicación para cada una de ellas.	

Tabla 6: Historia de usuario Realizar autoevaluaciones del tema "Teoría de Grafos" (Elaboración propia).

Historia de Usuario	
Número: HU_8	**Usuario:** Estudiante
Nombre de historia: Realizar autoevaluaciones del tema "Teoría de Grafos".	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Osmel Mojena Dubet.	
Descripción: El usuario accede al sistema y resuelve los ejercicios propuestos por el profesor para el entrenamiento.	

Tabla 7: Historia de usuario Adicionar ejercicios del tema "Teoría de Grafos" (Elaboración propia).

Historia de Usuario	
Número: HU_10	**Usuario:** Profesor
Nombre de historia: Adicionar ejercicios del tema "Teoría de Grafos".	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Programador Responsable: Osmel Mojena Dubet.	
Descripción: El usuario diseña los ejercicios y los pone a disposición de los estudiantes para la evaluación de los mismos. Estos ejercicios serán de responder verdadero o falso, seleccionar respuestas correctas, ya sea única o múltiple, subir un fichero o un valor entero. Todas pueden incluir justificación del estudiante y una explicación del profesor en caso de ser necesaria.	

2.6 Estilo arquitectónico.

El *framework* de desarrollo web Django emplea una modificación del estilo arquitectónico Modelo-Vista-Controlador (MVC), llamada *Model–Template–View* (MTV), que sería Modelo-Plantilla-Vista, está forma de trabajar permite que sea pragmático.

Como plantea Infante-Montero (2012), para comprender como funciona el MTV de Django se debe tener en cuenta, su analogía con el MVC de la siguiente manera:

- El modelo en Django continúa siendo Modelo.

- La vista en Django pasa a llamarse Plantilla.
- El controlador en Django pasa a llamarse Vista.

A partir de lo planteado se asumirá el estilo arquitectónico MTV de Django del cual se explica su funcionamiento a continuación y se podrá observar en la ilustración 4:

1- El navegador web envía una solicitud

2- La vista interactúa con el modelo para obtener los datos

3- La vista llama a la plantilla

4- La plantilla muestra la respuesta a la solicitud del navegador

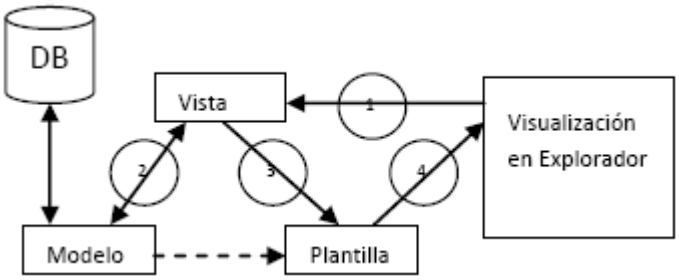


Ilustración 4: Flujo de trabajo del estilo arquitectónico Modelo-Vista-Plantilla (Condori, 2012).

- M significa "*Model*" (Modelo) que es el análogo de Modelo en MVC, la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos: cómo acceder a éstos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T significa "*Template*" (Plantilla) que es el análogo de Vista en MVC, la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación: cómo algunas cosas son mostradas sobre una página web u otro tipo de documento.
- V significa "*View*" (Vista) que es el análogo de Controlador en MVC, la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada: se puede pensar en esto como un puente entre los modelos y las plantillas.

Para cumplir con la arquitectura seleccionada, la implementación del módulo de Teoría de Grafos para el sistema interactivo y experimental cuenta con la siguiente estructura:

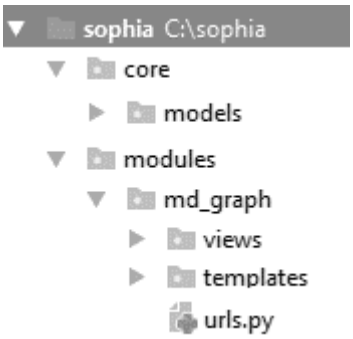


Ilustración 5: Estructura de directorios del módulo (Elaboración propia).

En la estructura de directorios anterior se pudo observar cómo se encuentran distribuidos en el sistema los elementos que conforman la arquitectura *MTV*, además se señala el fichero *urls.py*, este archivo es el encargado de hacer la petición a la vista correspondiente para cada *URL*[12] que interviene en el sistema.

2.7 Patrones de diseño.

Un patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede reutilizarse en diferentes configuraciones. El patrón no es una especificación detallada. Más bien, puede considerarla como una descripción de sabiduría y experiencia acumuladas, una solución bien probada a un problema común (Sommerville, 2011).

Patrones *GRASP.*

Los Patrones de Principios Generales para Asignar Responsabilidades (*GRASP* por sus siglas en inglés) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones.

Experto en Información: las responsabilidades deben ser asignadas a las clases que poseen la información para realizar dicha responsabilidad (Gómez, Duarte, y Guevara, 2014).

En la siguiente ilustración se muestra un ejemplo de cómo se pone de manifiesto el patrón experto, en este caso se usó en la clase del modelo *EvaluationQuiestion* para calcular la cantidad de preguntas que tiene una evaluación sin tener que implementar un método desde la vista (donde se encuentran las clases controladoras), ya que este modelo tiene la información necesaria para brindar toda la información acerca de cada uno de sus atributos.

[12] Localizador Uniforme de Recursos; secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.

```
def quantity_question(self):
    return self.evaluationquestion_set.count()
```

Ilustración 6: Código fuente para obtener la cantidad de preguntas de una evaluación (Elaboración propia).

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creadora. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización. La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia, es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien, el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización (Rodríguez, 2016).

Este patrón se pone de manifiesto en las vistas, donde se implementan las clases controladoras para crear objetos del modelo de datos, permitiendo acceder a estos de forma directa en cada una de las vistas para enviar la información necesaria hacia las plantillas, respondiendo a las peticiones del usuario a través de un navegador web.

```
@login_required(login_url='/')
@active_required(login_url='/lock/')
@permission_required(perm='core.change_evaluation', login_url=settings.ERROR_403_URL)
def view_evaluation(request, pk):
    evaluation = get_object_or_404(Evaluation, pk=pk, is_deleted=False)
    red = evaluation.subnet
    question = EvaluationQuestion.objects.filter(evaluation=evaluation).order_by('number')

    return render(request, 'md_graph/evaluation/view_evaluation.html', locals())
```

Ilustración 7: Código fuente para mostrar la información de una evaluación (Elaboración propia).

Alta Cohesión: una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Significa que las clases del sistema tienen asignadas solo las responsabilidades que les corresponde y mantienen una estrecha relación con el resto de las clases (Escalante, 2016).

Al usar este patrón se garantiza una mejor eficiencia en cuanto al tiempo de respuesta de las peticiones del usuario, en la siguiente ilustración se muestra un segmento de código que permite verificar si un grafo es regular, donde, para esto debe cumplir con una serie de condiciones triviales, que son implementadas en otras clases o métodos auxiliares, para futuras reutilizaciones, con esto se garantiza que no es necesario volver a implementar funcionalidades que dependan de otras condiciones que están previamente definidas.

En este caso para que el grafo sea regular tiene que cumplir con las condiciones de ser simple y el grado de todos los vértices sea el mismo, donde cada una de estas condiciones ya implementadas dependen de otras condiciones triviales, que, de igual manera ya están definidas para su utilización.

```
function check_regular() {
    if (VERTICES.length == 0) {
        showMessage('error', '', 'Debe introducir un grafo para experimentar.');
    }
    else {
        var r = $('#resp_regular').find('option:selected').val();
        if (r == -1) {
            has_empty('regular');
        }
        else {
            var result = es_regular();
            if (r == result) {
                has_success('regular');
            }
            else {
                has_error('regular');
            }
        }
    }
}
```

Ilustración 8: Código fuente para determinar si un grafo es regular (Elaboración propia).

Controlador: es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente una de las siguientes opciones. La respuesta puede ser mostrar una vista, ejecutar un método, devolver un mensaje, etc (Bermúdez, Jiménez, y Rodríguez, 2013). Se evidencia el uso de este patrón en el archivo *evaluation.py* que es el encargado de definir clases y funcionalidades que controlan las vistas que se muestran en la aplicación.

La imagen que se muestra a continuación muestra la vista que activa una evaluación, la cual cambia el estado de esta y muestra el mensaje “Evaluación activada.”

```
@login_required(login_url='/')
@active_required(login_url='/lock/')
@permission_required(perm='core.change_evaluation', login_url=settings.ERROR_403_URL)
def activate_evaluation(request, pk):
    evaluation = get_object_or_404(Evaluation, pk=pk, is_deleted=True)
    evaluation.is_deleted = False
    evaluation.deleted_at = None
    evaluation.deleted_by = None
    evaluation.save()
    messages.success(request, 'Evaluación activada.')
    return HttpResponseRedirect(reverse('modules:md_graph_list_evaluation'))
```

Ilustración 9: Código fuente para activar una evaluación una vez que haya sido eliminada (Elaboración propia).

Los patrones *GRASP* han posibilitado seguir una estructura de diseño única mediante la reutilización de los estándares que propone. Además, facilita la implementación a partir de los conocimientos ya existentes y permite no realizar búsquedas para solucionar los problemas que se presenten a medida que se diseña el sistema.

Patrones *GoF* (*Gang of Four*).

Representan patrones que dan soluciones técnicas basadas en POO[13] que favorecen la reutilización del código (Guerrero y Suárez, 2013).

Decorador: es un patrón estructural que extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase (Silva, 2007). Como ejemplo de este patrón se evidencia la utilización de los decoradores de Django o definidos por el programador, cómo es el caso de *active_required*:

```
@login_required(login_url='/')
@active_required(login_url='/lock/')
@permission_required(perm='core.change_evaluation', login_url=settings.ERROR_403_URL)
```

Ilustración 10: Decoradores para el control de sesiones y permisos (Elaboración propia).

[13] Programación orientada a objetos

2.8 Diagrama de Clases.

Un diagrama de clases en Lenguaje Unificado de Modelado (*UML*) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos (Bedoya, Hernández y Villegas, 2016).

Los diagramas de clases contienen normalmente los siguientes elementos, clases, interfaces, colaboraciones, relaciones de dependencia, generalización y asociación. Estos se utilizan para modelar la vista estática de un sistema, muestra principalmente los requisitos funcionales de un sistema y los servicios que el sistema debe proporcionar a sus usuarios finales.

Las clases se representan por rectángulos que muestran el nombre de la clase y opcionalmente el nombre de las operaciones y atributos. Los compartimientos se usan para dividir el nombre de la clase, atributos y operaciones. Adicionalmente las restricciones, valores iniciales y parámetros se pueden asignar a clases, ver anexo C.

2.9 Diagrama de clases del diseño.

Un Diagrama de Clases de Diseño muestra la especificación para las clases de software de una aplicación. Incluye la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias.

Un Diagrama de Clases de Diseño muestra definiciones de entidades software más que conceptos del mundo real.

Conallem (1999) definió una extensión a la que denominó *WAE (Web Application Extension)* para *UML*, la cual cuenta con un conjunto de estereotipos que se pueden asociar a las clases y establecer las relaciones entre estas para representar una aplicación web, los cuales se pueden observar en la siguiente tabla:

Tabla 8: Estereotipos web para UML

Estereotipos para las clases	
Estereotipo	**Descripción**
 *Client Page*	Representan páginas que son dibujadas por el navegador web y pueden ser una combinación de algún o algunos lenguajes de marcado, *scripts* del lado del cliente, islas de datos, etc.
 *Server Page*	Representa una página web que tiene *scripts* ejecutados por el servidor. Estos *scripts* interactúan con los recursos que se encuentran al alcance del servidor. Sólo puede mantener relaciones con objetos que se encuentren en el servidor.
 *Form*	Representa una colección de campos de entrada que forman parte con una página del lado cliente (*Client Page*). Tiene una correspondencia directa con la etiqueta <*FORM*> de *HTML*.
Estereotipos para las Relaciones entre las Clases	
Link	Representa un apuntador desde una "*client page*" hacia una "*client page*" o "*server page*". Corresponde directamente con una etiqueta <a> de *HTML*
Submit	Esta relación siempre se da entre una "*form*" y una "*server page*", por supuesto, la "*server page*" procesa los datos que la "*form*" le envía (*submits*)
Build	Sirve para identificar cuales "*server page*" son responsables de la creación de una "*client page*". Una "*server page*" puede crear varias "*client page*", pero una "*client page*" sólo puede ser creada por una sola "*server page*". Esta relación siempre es unidireccional
Redirect	Esta es también una relación unidireccional que indica que una página web redirige hacia otra. En caso de que la página origen sea una "*client page*" esta asociación corresponderá con la etiqueta "*META*" y valor

HTTP-EQUIV de "*Refresh*".

A continuación, se muestran dos (2) diagramas de clases de las historias de usuario: Adicionar ejercicios del tema "Teoría de Grafos" y Resolver ejercicios del tema "Teoría de Grafos".

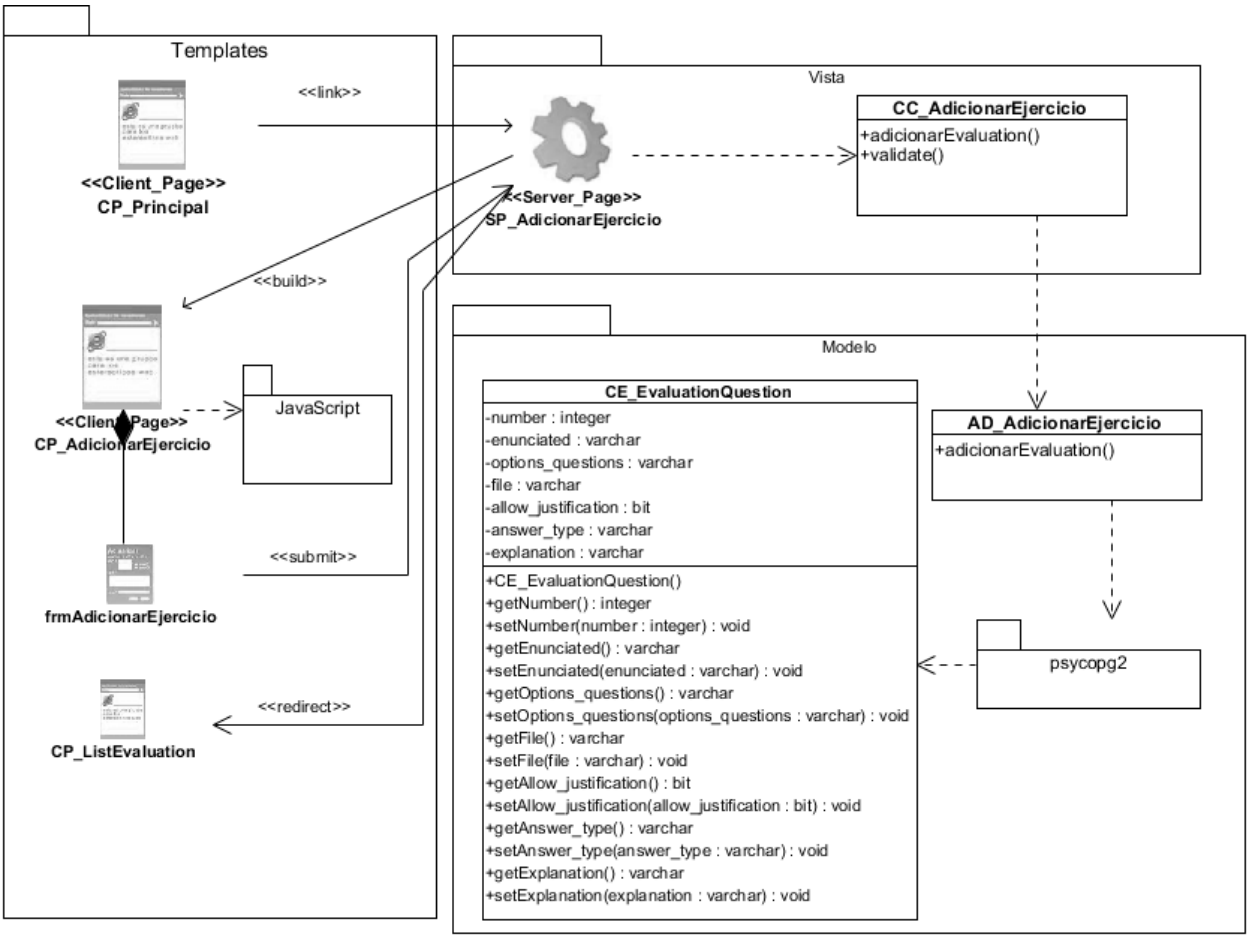


Ilustración 11: Diagrama de clases del diseño de la HU Adicionar ejercicios del tema "Teoría de Grafos" (Elaboración propia).

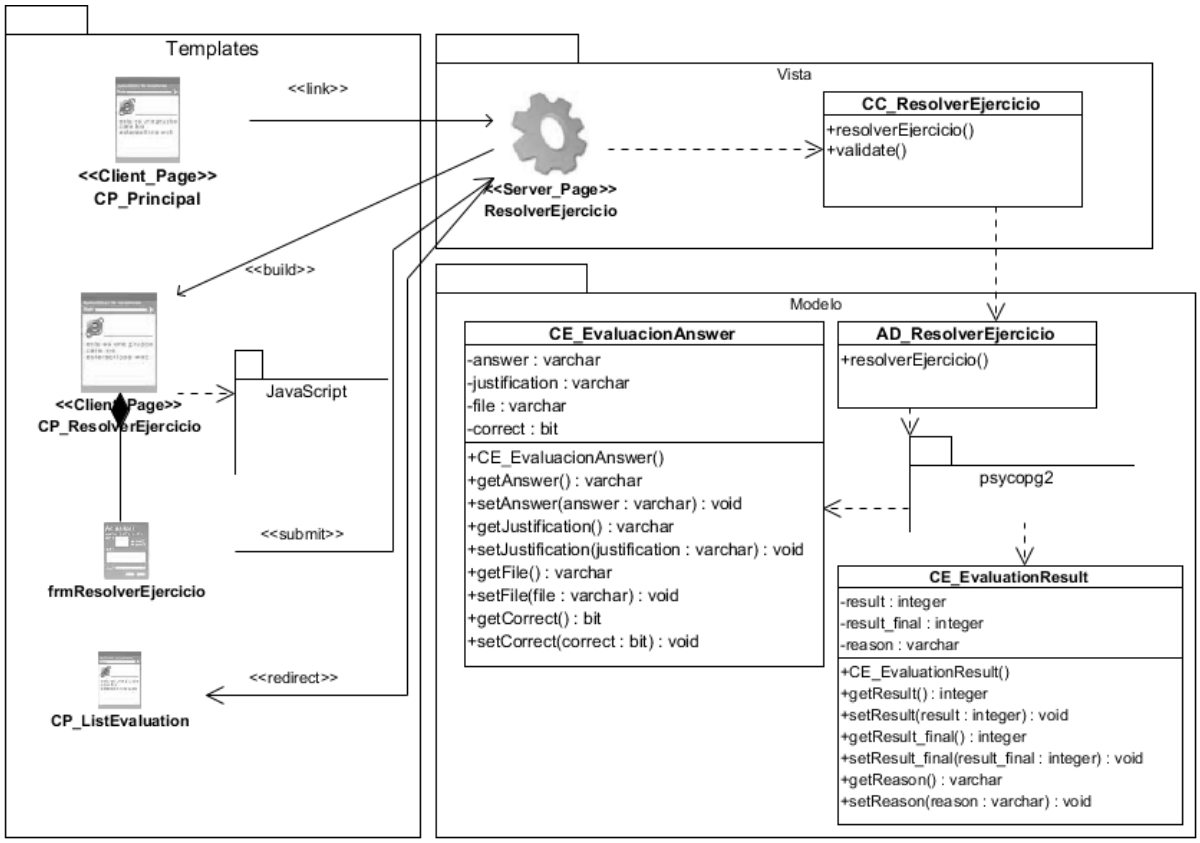


Ilustración 12: Diagrama de clases del diseño de la HU Realizar competencias del tema "Teoría de Grafos" (Elaboración propia).

2.10 Modelo de datos.

Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos o podríamos decir que es un conjunto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos (Redondo, 2017).

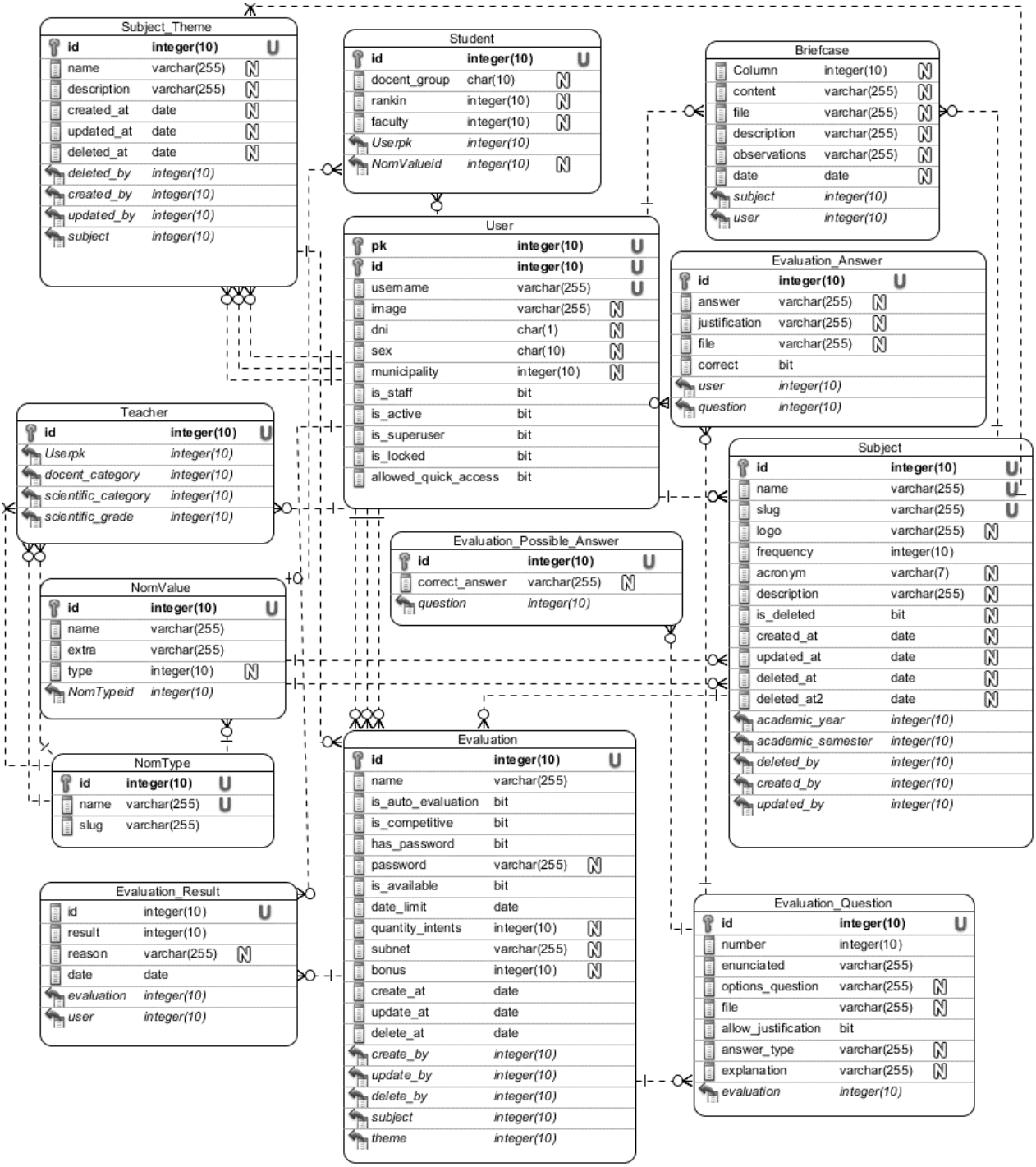


Ilustración 13: Modelo de datos (Elaboración propia).

En la ilustración anterior se muestra la representación de modelo de datos donde se establecen todas las relaciones entre los estudiantes los profesores y las evaluaciones.

2.11 Modelo de despliegue.

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene memoria y a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representan un procesador o un dispositivo sobre el que se pueden desplegar los componentes. La relación entre nodo y componente que despliega puede mostrarse con una relación de dependencia (Prefacio, 2000). La ilustración representa la vista de despliegue del sistema.

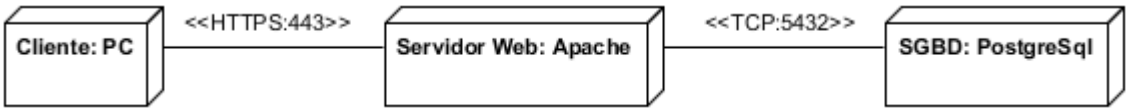


Ilustración 14: Diagrama de despliegue (Elaboración propia).

PC Cliente: Se refiere a las estaciones de trabajo que realizan las peticiones al servidor de aplicaciones donde está hospedada la plataforma que soporta el módulo Teoría de Grafos del sistema interactivo y experimental para el proceso de enseñanza-aprendizaje de la Matemática Discreta, mediante un navegador web utilizando el protocolo de comunicación *HTTPS* por el puerto 443.

Servidor Web: Es el encargado de brindar la interfaz de la plataforma para que los usuarios puedan hacer uso de esta, almacena todo el código fuente del sistema y se comunica por medio de los protocolos *TCP* con el servidor de bases de datos.

Sistema Gestor de Bases de Datos: Almacena toda la información que brinda la plataforma hospedada en el servidor de aplicaciones. La información es obtenida o modificada en dependencia del nivel de privilegio del usuario que realiza la petición. La comunicación con el servidor de aplicaciones es a través del protocolo *TCP* empleando el puerto 5432.

2.12 Conclusiones parciales.

El levantamiento de las funcionalidades y sus respectivas historias de usuario, en conjunto con los requisitos no funcionales y demás artefactos permitió diseñar una propuesta de solución que incluye una serie de patrones *GRASP* y *GoF* a utilizar en la implementación para lograr una adecuada reutilización del código para futuras versiones. Además, se definieron las estrategias de gamificación para lograr una mejor motivación en los estudiantes. La definición del estilo arquitectónico y la realización de los diagramas de clases del diseño permitieron un mejor entendimiento del funcionamiento del sistema.

CAPÍTULO 3. Implementación y pruebas de la propuesta de solución

3.1 Introducción

En el presente capítulo se muestra el diagrama de componentes como resultado del diseño desarrollado. La fase de implementación comprende la materialización, en forma de código, de los artefactos y las descripciones con el objetivo de conformar el producto final requerido por el cliente. Además, en este capítulo se presenta la validación del sistema, con el objetivo de corroborar la correspondencia entre el producto y los requisitos definidos anteriormente.

3.2 Diagrama de componentes

El diagrama de componentes muestra las relaciones estructurales entre los componentes de un sistema. Los componentes se consideran unidades autónomas encapsuladas dentro de un sistema o subsistema que proporcionan una o más interfaces. Estos diagramas son generalmente dirigidos al personal de aplicación de un sistema, además presenta una comprensión temprana del sistema global que se está construyendo (Bell, 2004).

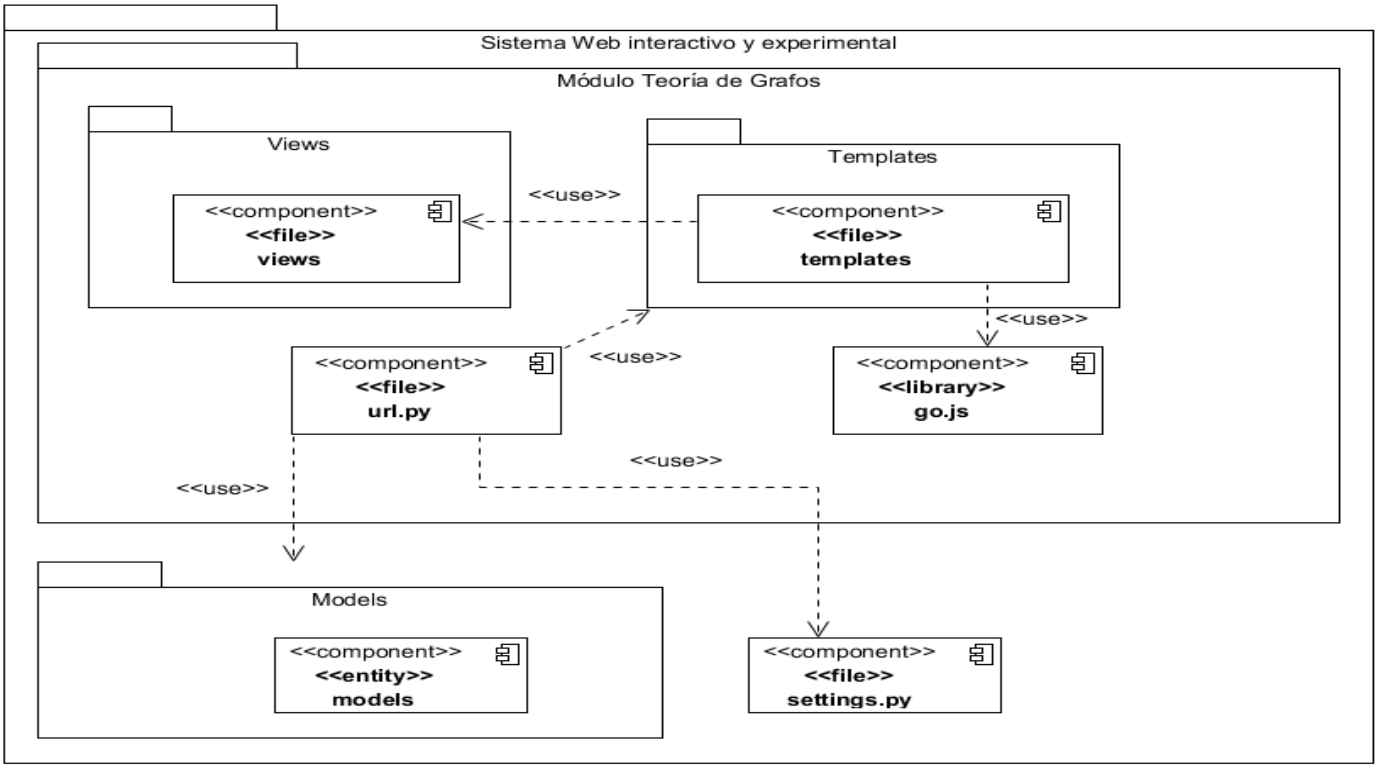


Ilustración 15: Diagrama de componentes

En el diagrama anterior muestra los componentes que componen el módulo de Teoría de Grafos, donde los componentes *views*, *templetes* y *model*, contienen todos los ficheros que forman parte del mismo, los cuales se pueden observar en el diagrama del anexo B. A continuación, se describen cada uno de los ficheros que componen los componentes del módulo de Teoría de Grafos

Tabla 9: Descripción de los ficheros que componen los componentes

	Ficheros	Descripción
Templetes	*md_main.html*	Este fichero muestra la página principal del módulo, la cual permite el acceso a las diferentes unidades didácticas del mismo.
	content.html	Este fichero muestra el contenido teórico del tema.
	graphs_experimentation.html	Este fichero muestra un conjunto de operaciones que se realizan sobre los grafos, permite resolver cada una de ellas, además de poder generarlos empleando la librería *go.js*.
	list_eval.html	Este componente muestra las evaluaciones disponibles en el sistema.
	list_group.html	Este fichero muestra los grupos del profesor que está autenticado en el sistema.
	resolve_evaluation.html	Este fichero muestra la evaluación seleccionada para resolver.
	review_eval.html	Este fichero muestra al profesor la lista de los estudiantes de un grupo y de una evaluación seleccionada.
	evaluation_estudent_view.html	Este fichero muestra al profesor la respuesta de la evaluación que dio el estudiante a cierta evaluación.
	add_evaluation.html	Este fichero muestra los parámetros necesarios para agregar una nueva evaluación al sistema.

	view_evaluation.html	Este fichero muestra los datos de la evaluación seleccionada y permite añadir ejercicios a dicha evaluación.
	list_evaluation.html	Este fichero muestra todas las evaluaciones registradas en el sistema.
	add_exercise.html	Este fichero muestra los parámetros necesarios para agregar ejercicios a las evaluaciones.
Views	*evaluate.py*	Este fichero es el encargado de enviar a las plantillas todas las evaluaciones disponibles en el sistema y las estadísticas de los resultados en las evaluaciones.
	resolve.py	Este fichero es el encargado de enviar a la plantilla todos los resultados relacionados con las evaluaciones.
	views.py	Este fichero es el encargado de redirigir las plantillas que solo trabajan del lado del cliente.
	exercise.py	Este fichero es el encargado de realizar las operaciones de inserción, actualización y eliminación de los ejercicios.
	evaluation.py	Este fichero es el encargado de realizar las operaciones de inserción, actualización y eliminación de las evaluaciones.
Models	*Evaluation.py*	Clase entidad que almacena todas las evaluaciones del sistema.
	EvaluationQuestion.py	Clase entidad que almacena los ejercicios de cada una de las evaluaciones.
	EvaluationPossibleAnswer.py	Clase entidad que almacena las posibles respuestas de las evaluaciones registradas en el sistema.
	EvaluationAnswer.py	Clase entidad que almacena las respuestas de las evaluaciones dadas por los estudiantes.
	EvaluationResult.py	Clase entidad que almacena los resultados de cada estudiante de las evaluaciones resueltas.

Briefcase.py	Clase entidad que almacena los grafos creados en la experimentación.
Student.py	Clase entidad que almacena los datos de los estudiantes registrados en el sistema.
Tacher.py	Clase entidad que almacena los datos de los profesores almacenados en el sistema.

3.3 Estándares de codificación

Cada programador tiene su propia formar de escribir los códigos, pero de la forma que se use depende la facilidad de que otros programadores entiendan el código y se les facilite su reutilización, de ahí se desprende la importancia de los estilos de programación, también conocidos como estándares o convenciones de código los cuales definen un grupo de convenciones para escribir código fuente en ciertos lenguajes de programación. A continuación, se relacionan de estándares de codificación a utilizar en la implementación del sistema (Abraira, 2016).

A continuación, se especifican los estándares de codificación a utilizar en la construcción de la solución:

Tabla 10: Estándares de codificación

Tipo de estándar	Descripción
Alineación	❖ Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes). ❖ Utilizar una indentación de una tabulación para cada línea con excepción de la primera. ❖ La indentación se realizará solamente con tabulaciones, no deben utilizarse nunca los cuatro (4) espacios.
Máxima longitud de las líneas	❖ Todas las líneas deben estar limitadas a un máximo de setenta y nueve (79) caracteres. ❖ Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las líneas largas. ❖ En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas.

Líneas en blanco

- Separar las funciones de alto nivel y definiciones de clases con dos (2) líneas en blanco.
- Las definiciones de métodos dentro de una clase deben separarse por una (1) línea en blanco.
- Se pueden utilizar líneas en blanco, escasamente, para separar secciones lógicas.

Codificaciones

- Utilizar la codificación UTF-8.
- Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”.

Importación

- Las importaciones deben estar en líneas separadas.
- Siempre deben colocarse al comienzo del archivo.
- Deben quedar agrupadas de la siguiente forma:
  1. Importaciones de la librería estándar.
  2. Importaciones terceras relacionadas.
  3. Importaciones locales de la aplicación / librerías.
- Cada grupo de importaciones debe estar separado por una línea en blanco.

Espacios en blanco en expresiones y sentencias

- Evitar utilizar espacios en blanco en las siguientes situaciones:
  1. Inmediatamente dentro de paréntesis, corchetes y llaves.
  2. Inmediatamente antes de una coma, un punto y coma o dos puntos.
  3. Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función.
  4. Inmediatamente antes de un corchete que empieza una indexación.
  5. Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.

	❖ Deben rodearse con exactamente un espacio los siguientes operadores binarios: 1.Asignación (=). 2. Asignación de aumentación (+=, -=, etc.). 3. Comparación (==, <, >, >=, <=, !=, <>, in, not in, is, is not). 4. Expresiones lógicas (and, or, not). ❖ Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad. ❖ No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.
Comentarios	❖ Los comentarios deben ser oraciones completas. ❖ Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula. ❖ Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc. ❖ Si un comentario es corto el punto final puede omitirse.
Comentarios en bloque	❖ Deben estar indentados al mismo nivel que el código a comentar. ❖ Cada línea de un comentario en bloque comienza con un numeral (#) y un espacio en blanco.
Comentarios en la misma línea	❖ Se recomienda utilizarlos escasamente. ❖ Se debe definir comenzando por un numeral (#) seguido de un espacio en blanco. ❖ Deben ubicarse en la misma línea que se desea comentar.

Cadenas de documentación	❖ Deben quedar documentados todos los módulos, funciones, clases y métodos públicos. ❖ Para definir una cadena de documentación debe quedar encerrada dentro de (“””). ❖ Los (“””) que finalizan una cadena de documentación deben quedar en una línea a no ser que la cadena sea de una sola línea.
Convenciones de nombramiento	❖ Nunca se deben utilizar como simple caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de los números uno (1) y cero (0). ❖ Los módulos deben tener un nombre corto y en minúscula. ❖ Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas). ❖ Los nombres de las excepciones deben estar escrito también en la convención “CapWords” utilizando el sufijo “Error”. ❖ Los nombres de las funciones deben estar escritos en minúscula separando las palabras con un guión bajo “_”. ❖ Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (_).

3.4 Pruebas de software

El proceso de pruebas se centra en los procesos lógicos internos del software, asegurando que todas las sentencias se han comprobado, y en los procesos externos funcionales, es decir, la realización de las pruebas para la detección de errores. Además, son utilizadas para identificar posibles fallos de implementación, calidad o usabilidad de un programa (Pressman, 2010).

3.4.1. Pruebas de integración

Este tipo de pruebas consiste en la comprobación de que los elementos del software que interactúan entre sí, funcionan de manera correcta. Es una forma de chequear la correcta interrelación de los distintos componentes del sistema. En el caso de la solución desarrollada es la verificación de una correcta interoperabilidad entre el módulo desarrollado y el sistema web interactivo y experimental para el proceso de enseñanza-aprendizaje de la Matemática Discreta.

El sistema Interactivo y experimental para el cual se desarrolla el módulo de Teoría de Grafos, permite instalar los paquetes (módulos) importándolos en formato "*.zip*", donde dicho sistema extrae todos los componentes y los copia en los directorios correspondientes, donde solo es necesario garantizar el vínculo entre el sistema y la URL principal del módulo, el resto de las responsabilidades están en el módulo, quien garantiza un correcto funcionamiento.

Al integrar el módulo de Teoría de Grafos al sistema web interactivo y experimental este se integró correctamente sin presentar ninguna no conformidad, por tanto, se concluye que existe una correcta integración entre los componentes internos del sistema.

3.4.2. Pruebas funcionales

Se denominan pruebas funcionales *o Functional Testing*, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplen con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales (Guerrero, 2017).

A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, ya que los *testers* o analistas de pruebas no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas (Serna, Bedoya, y López, 2015).

Se realizó el diseño del caso de prueba para validar la propuesta de solución implementada con el objetivo de comprobar que la herramienta agregue correctamente una evaluación en el sistema.

A continuación, se muestra la especificación del caso de prueba para el requisito "Añadir Evaluación".

Tabla 11: Variables empleadas en el caso de prueba para el escenario Añadir Evaluación

No.	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	Disponibilidad	Casilla de verificación	No	Valor verdadero o falso
2	Asignatura	Campo de selección	No	Contiene letras y números
3	Tema	Campo de selección	SÍ	Contiene los temas de la asignatura
4	Nombre	Campo de texto	No	Contiene solo letras
5	Fecha límite	Campo de texto	No	Números y caracteres alfanuméricos
6	Cantidad de intentos	Campo numérico	No	Contiene solo números positivos
7	Protegido con contraseña	Casilla de verificación	No	Valor verdadero o falso
8	Contraseña	Campo de texto	SÍ	Contiene letras, números y caracteres alfanuméricos
9	Repetir contraseña	Campo de texto	SÍ	Contiene letras, números y caracteres alfanuméricos
10	Especificación de subred	Campo de texto	SÍ	Contiene letras, números y caracteres alfanuméricos
11	Bonificación	Campo numérico	No	Contiene solo números positivos

Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.

Tabla 12: Caso de prueba Añadir Evaluación (parte 1)

Escenario	Descripción	V1	V2	V3	V4	V5	V6	Respuesta	Flujo central
EC 1.1 Añadir una evaluación datos correctos	Se registran los datos de una evaluación correctamente	V SÍ	V Nombre de la asignatura	N/A Nombre del tema de la asignatura	V Nombre de la evaluación	V Fecha de expiración	V Cuantos intentos tiene la evaluación	Muestra el mensaje “Evaluación insertada.”	El profesor rellena los campos de forma correcta y da clic en el botón Guardar
EC 1.2 Añadir una evaluación datos incorrectos	Existen campos obligatorios vacíos	V SÍ	- Vacío	V Nombre del tema de la asignatura	- Vacío	- Vacío	V Cuantos intentos tiene la evaluación	Muestra el mensaje “Rellene los campos obligatorios”	El profesor rellena los campos dejando algunos obligatorios vacíos

Tabla 13: Caso de prueba Añadir Evaluación (parte 2)

Escenario	Descripción	V7	V8	V9	V10	V11	Respuesta	Flujo central
EC 1.1 Añadir una evaluación datos correctos	Se registran los datos de una evaluación correctamente	N/A SÍ	N/A Contraseña	N/A Repetir contraseña	N/A Subredes que pueden acceder	V Bonificación	Muestra el mensaje “Evaluación insertada.”	El profesor rellena los campos de forma correcta y da clic en el botón Guardar
EC 1.2 Añadir una evaluación datos incorrectos	Existen campos obligatorios vacíos	V No	V Vacío	V Vacío	V Subredes que pueden acceder	V Bonificación	Muestra el mensaje “Rellene los campos obligatorios”	El profesor rellena los campos dejando algunos obligatorios vacíos

Resultados de las pruebas funcionales.

Las pruebas funcionales se realizaron en dos iteraciones donde se aplicaron los casos de prueba diseñados, la primera iteración arrojó diecisiete (17) no conformidades y la segunda iteración arrojó dos (2) no conformidades. A continuación, se muestran los resultados de las pruebas en cada iteración.

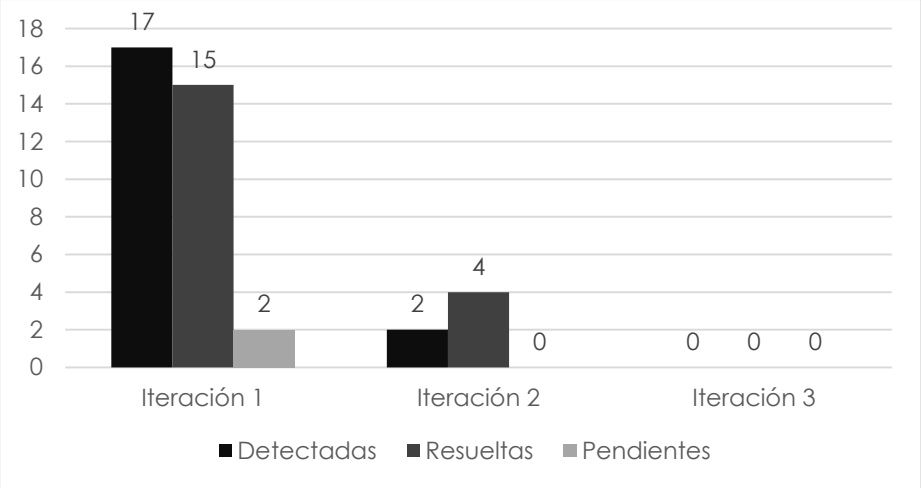


Ilustración 16: Gráfico de resultado de Pruebas funcionales

El gráfico de la ilustración anterior muestra el comportamiento de las no conformidades encontradas durante el proceso de prueba, donde se observa que en la primera iteración se detectaron diecisiete (17) no conformidades, de ellas cuatro (4) errores de interfaz, nueve (9) validaciones incorrectas y cuatro (4) errores de excepción, quedando pendientes para la siguiente iteración dos (2) de interfaz.

En la segunda iteración se detectaron dos (2) nuevas no conformidades, las cuales se resolvieron satisfactoriamente además de las dos (2) pendientes de la iteración anterior.

Luego de una tercera iteración no se detectaron nuevas no conformidades, por lo que el módulo se encuentra listo para su funcionamiento.

3.4.3. Pruebas de rendimiento.

Pruebas de carga.

La carga de trabajo se refiere a la capacidad máxima que tiene un servidor web, para atender a un conjunto de usuarios de manera simultánea. Por ello, las actividades de esta etapa tienen relación con comprobar, de manera anticipada, el funcionamiento que tendrá el servidor del Sitio Web cuando esté en plena operación (Díaz y Dosting, 2014).

Las pruebas en este caso consisten en simular una carga de trabajo similar y superior a la que tendrá cuando el sitio esté funcionando, con el fin de detectar si el software instalado cumple con los requerimientos de muchos usuarios simultáneos y también si el hardware es capaz de soportar la cantidad de visitas esperadas (Serna, Bedoya, y López, 2015).

Pruebas de estrés.

Enfocada a evaluar cómo el sistema responde bajo condiciones anormales. (Extrema sobrecarga, insuficiente memoria, servicios y hardware no disponible, recursos compartidos no disponible (Pérez y Paumier, 2015).

Resultados de las pruebas de carga y estrés.

Con el objetivo de saber con un alto nivel de proximidad el soporte del módulo de Teoría de Grafos, teniendo en cuenta que la matrícula del primer año de la carrera oscila sobre los 1000 estudiantes, se desarrollaron las pruebas para un total de 700,1000 y 3000 usuarios concurrentes.

Las pruebas se desarrollaron con el apoyo de la herramienta Apache Meter 2.12, en la que se simuló el entorno donde debe interactuar el módulo para obtener la información más correcta acerca del comportamiento y resultados en general. Por lo que fue elegido un ambiente de gama con las siguientes características:

Software:

- Sistema Operativo: Windows 10 Arquitectura de 64bits
- Sistema Gestor de Bases de Datos: PostgreSQL 9.4
- Servidor Web: Apache v2.4
- Máximo de hilos concurrentes: entre 700 y 1000

Hardware:

- Microprocessor: Intel Core(TM) i3 – 2120M CPU @ 3.30 GHz
- Memoria RAM: 4GB
- Tarjeta de Red: Ethernet 10/100 Mbps

Se describen a continuación las variables que miden los resultados de las pruebas.

Cantidad de Hilos (Cant de Hilos): cantidad de usuarios que se conectan concurrentemente a la aplicación.

#Muestras: cantidad de peticiones realizadas para cada URL.

Media: tiempo promedio en milisegundos en que se obtiene los resultados.

Min: tiempo mínimo que demora un hilo en acceder a una página.

Max: tiempo máximo que demora un hilo en acceder a una página.

%Error: porciento de error de las páginas que no llegan a cargar de manera correcta.

Rendimiento: el rendimiento se mide en cantidad de solicitudes por segundo.

Kb/sec: velocidad de carga de las páginas**.**

La siguiente tabla muestra el resultado que arrojó la prueba para un total de 700, 1000 y 3000 usuarios concurrentes, donde el sistema responde en un tiempo promedio de 1.5, 1.1 y 1.0 segundos respectivamente.

Tabla 14: Resultados de las pruebas de carga y estrés.

Cant de Hilos	#Muestras	Media	Min	Máx	*%Error*	*Rendimiento*	*Kb/s*
700	1003	1542	45	3122	0.00	89.14	1003
1000	2600	1145	140	4255	0.01	78.25	1500
3000	2800	1002	210	4756	0.27	67.25	2100

3.5 Contribución al proceso de enseñanza aprendizaje de la Teoría de Grafos

Con el objetivo de evaluar los efectos del módulo desarrollado en el aprendizaje de la Teoría de Grafos, se utilizó en la asignatura Matemática Discreta, de forma total en el grupo FI08 y de forma parcial en los demás grupos docentes de la Facultad Introductoria de Ciencias Informáticas.

Además, fue utilizado por los profesores para sus preparaciones metodológicas. Vale señalar que se realizó una entrevista a varios de los principales directivos del proceso docente en la facultad y universidad los cuales mostraron su grado de satisfacción, además algunos de ellos otorgaron avales a esta investigación (ver anexo A).

Resultados de la entrevista realizada a directivos del proceso docente

Se entrevistaron a 5 directivos tanto metodológicos como administrativos mediante una guía de entrevista diseñada con el propósito de evaluar el impacto del software desarrollado en el proceso de enseñanza-aprendizaje de la Teoría de Grafos. Las dimensiones desarrolladas en la entrevista fueron:

I. **Dimensión motivacional**. Con el objetivo de evaluar los efectos del software en la motivación de los estudiantes.

II. **Dimensión docente**. Con el objetivo de determinar los aportes del software al proceso de enseñanza-aprendizaje de la Teoría de Grafos.

III. **Dimensión cognitiva**. Con el objetivo de identificar los aportes al rendimiento académico de los estudiantes.

Los principales resultados de la entrevista realizada fueron los siguientes:

- Aumentó la participación de los estudiantes en las clases y su actividad de estudio individual, lo que se traduce en una mayor motivación en su proceso de aprendizaje.
- El sistema permitió concretar la enseñanza de la Teoría de Grafos con la posibilidad de experimentar con los parámetros del contenido.
- De especial importancia es la posibilidad que se brinda de poseer una cantidad de ejercicios ilimitados, donde los estudiantes proponen sus propios ejemplos.
- Se resalta el tema de la autoevaluación, donde cada estudiante puede proponerse evaluaciones y ver los resultados de la misma siempre con una retroalimentación, lo cual es primordial en el proceso de enseñanza-aprendizaje.
- Es una de las propuestas que es pionera en abordar el tema de la gamificación, a través del desarrollo de competencias a través de un ranking centralizado, lo cual motiva al estudiante para enfrentarse a contenidos de alta complejidad.
- El material posee un diseño agradable, fácil de manejar y de una excelente factura.
- Se propicia la posibilidad de que el profesor trabaje las diferencias individuales en la clase. Además de la diferenciación del estudio independiente, con amplias posibilidades de desarrollar la interacción estudiante-profesor.

Efectos en el rendimiento académico

El sistema fue utilizado por los profesores en algunas de sus clases. No obstante, el grupo FI08 lo utilizó en la totalidad de sus clases. De esa forma este grupo, como se pudo comprobar en el Sistema de Gestión Universitaria, obtuvo los mejores resultados de promoción en el examen de Teoría de Grafos entre los 40 grupos de la facultad, con un 95%, es decir, solo un estudiante suspendió dicho examen. Vale señalar además que este grupo obtuvo un 65% de calidad en la nota, los mejores resultados de la Universidad.

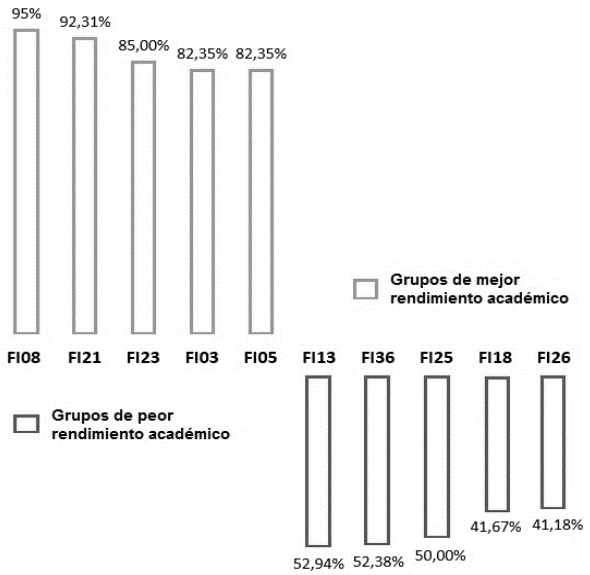


Ilustración 17: Mejores y peores grupos de la universidad en relación al rendimiento académico.

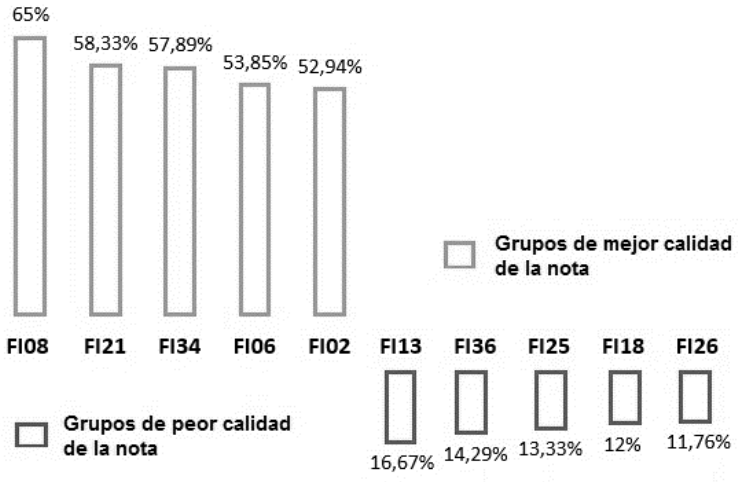


Ilustración 18: Mejores y peores grupos de la universidad en relación a la calidad de la nota

3.6 Conclusiones del capítulo

La realización del diagrama de componentes, así como la especificación de los estándares de codificación permitieron visualizar con facilidad la estructura general de la solución y permitieron implementar códigos completamente legibles para el buen entendimiento de otros programadores facilitando así su recodificación. Por último, la detección temprana de errores, mediante la aplicación de pruebas permitió validar que el sistema funciona como inicialmente fue diseñado, que no se incurrió en la violación de ningún requisito y la implementación fue correcta. Es importante concluir que el módulo desarrollado tuvo gran impacto en el proceso de enseñanza aprendizaje de la Teoría de Grafos, propiciando una mayor motivación de los estudiantes y por ende un mejor rendimiento académico.

CONCLUSIONES GENERALES

La presente investigación tuvo como base el desarrollo de un módulo de Teoría de Grafos para el apoyo al proceso de enseñanza-aprendizaje de la Matemática Discreta en la carrera Ingeniería en Ciencias Informáticas, para ello se dio cumplimiento a una serie de objetivos específicos, los cuales fueron cumplidos satisfactoriamente, por lo que se puede arribar a las siguientes conclusiones:

- La profundización en los principales supuestos teóricos que sustentan el desarrollo y utilización de un módulo de Teoría de Grafos en el proceso de enseñanza-aprendizaje de la MD en la Ingeniería en Ciencias Informáticas permitió ratificar a la Teoría de Grafos como un tema esencial en la formación de profesionales de la informática.
- El análisis y diseño del módulo de Teoría de Grafos permitió identificar los recursos a incluir en el sistema teniendo en cuenta el grado de complejidad de impartición y asimilación del contenido, la posibilidad de representación gráfica y la posibilidad de experimentar. Lo anterior se materializó en una propuesta de solución donde se integran la unidad de información, didáctica y de evaluación y retroalimentación; para lo cual se generaron una serie de artefactos ingenieriles.
- El módulo desarrollado fue utilizado por estudiantes y profesores de la Facultad Introductoria de Ciencias Informáticas, con excelentes resultados en la motivación, el rendimiento académico y en la preparación metodológica. De forma general se concluye que contribuyó de forma exitosa a la autoevaluación y la interacción estudiante-profesor en el proceso de enseñanza-aprendizaje de la Teoría de Grafos.

RECOMENDACIONES

El autor de la presente investigación propone las siguientes recomendaciones:

- Realizar versiones posteriores donde el módulo permita la experimentación con grafos dirigidos y ponderados.
- Incluir nuevas estrategias de gamificación en aras de continuar fortaleciendo la motivación de los estudiantes al estudio de la Matemática Discreta.

REFERENCIAS

Abraira, J. A. (2016). Trabajo de Diploma. *Sistema interactivo - experimental para el proceso de enseñanza – aprendizaje de la Matemática Discreta en la carrera Ingeniería en*.

Gutiérrez, E. (2009). JavaScript: Conceptos básicos y avanzados. Barcelona, España: Informática Técnica.

Apache Software Foundation. (20 de mayo de 2011). *The Apache HTTP Server Project.* Recuperado el 6 de diciembre de 2016, de http://httpd.apache.org/

Arteta, N. (5 de enero de 2015). *Hipertextual*. Obtenido de https://hipertextual.com/archivo/2015/01/que-es-gamificacion/

Bedoya, P. A., Hernández, H. D., y Villegas, D. A. (2016). Sistema de automatización para gestión de procesos administrativos y operativos. Universidad Tecnológica de Pereira.

Bell, D. (2004). *UML basics: The component diagram.* Obtenido de: https://www.ibm.com/developerworks/rational/library/dec04/bell/

Bermúdez, G. S., Jiménez, I. M., y Rodríguez, L. R. (2013). Uso de Patrones de diseño: Un caso Práctico. *Journal of Tropical Engineering 22.2*.

Bosch, M., y Gascón, J. (2013). *LAS PRÁCTICAS DOCENTES DEL PROFESOR DE MATEMÁTICAS.* Barcelona.

Canós, J., Letelier, P., y Penadés, M. C. (2003). Metodologías Ágiles en el desarrollo de Software. VAlencia.

Caraballo, S., y Cicata, R. (2012). *Hacia una didáctica de la Informática.*

Churchill, W. (2007). *Towards a useful classification of learning objects.* Educational Technology Research y Development.

CMS. (27 de marzo de 2008). CMS. Selecting a development approach. Recuperado el 9 de diciembre de 2016, de http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

Conallem, J. (1999). *Building Web Aplication with UML (2nd Edition).*

Condori Ayala, J. L. (2012). Phython - Django Framework de desarrollo web para perfeccionistasBasado en el Modelo MTV. *Revista de Información, Tecnología y Sociedad*.

REFERENCIAS

Condori, J. L. (2012). Python-Django. Framework de desarrollo web para perfeccionistas basado en el. 36. Recuperado el 9 de diciembre de 2016

Cortés, M. I., Hernández, C. R., y Cabrera, M. M. (2017). Sistema informático para la administración de riesgos en proyectos. *Universidad y Sociedad*.

Cortés, V. P., Bru, J. M., & Villalobos, A. R. (2016). An application of Graph Theory to improve the planning of work routes for a company in the vending sector. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, 7.

Díaz, J. A., y Dosting, A. (2014). *Gestión de empresas de desarrollo de software a partir de aplicaciones de tipo código abierto.*

Díaz, V. M. (2015). La gamificación educativa. Una alternativa para la enseñanza creativa. *Digital Education Review.*

Dubinsky, E. (2013). *Revista Latinoamericana de Educación en Matemática Educativa*, 47.

Ecured. (28 de junio de 2017). Obtenido de https://www.ecured.cu/Requisitos_no_funcionales

Escalante, L. C. (2016). Aplicación de Patrones de Diseño para Garantizar Alta Flexibilidad en el Software. Tecnología y Desarrollo.

Espinosa, R. S. (2016). Presentación. Juegos digitales y gamificación aplicados en el ámbito de la educación. *RIED: Revista Iberoamericana de Educación a Distancia*, 27-33. Obtenido de http://www.educativa.com

Galvín, F., y Javier, F. (2016). *Aspectos algebraicos en Teoría de Grafos.*

Gauchat, J. D. (2014). *El gran libro de HTML5.* Editorial Tomera Ltda.

Gómez, A. R., Duarte, A. Q., y Güevara, C. D. (2014). Desarrollo ágil de software aplicando programación extrema. *Revista Ingenio UFPSO 5.1,* 25.

González, C. S., y Carreño, A. M. (2015). Técnicas de gamificación aplicadas en la docencia de Ingeniería Informática. *ReVision 8.1.*

GoJS. (2017). GoJS. Interactive JavaScript Diagrams in HTML: https://gojs.net.

Grey, L. W., y Viltres, Y. M. (2015). Proceso de mejora del Sistema de Gestión de Proyectos para Cuba y Venezuela. *Campus Virtuales 3.2*.

REFERENCIAS

Guadalupe, M., Becerra, L. E., y Noriega Paltan, N. G. (2016). *Los indicadores esenciales de evaluación en el aprendizaje del área de matemática, en el cuarto año de educación básica de la unidad educativa "juan de velasco" riobamba, período 2014-2015.* Obtenido de http://dspace.unach.edu.ec/handle/51000/2388

Guerrero, C. A., y Johanna M. Suárez, L. E. (2013). Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*.

Guerrero, L. P. (2017). Tecnología Investigación y Academia. *Gestión en Proyectos de Software*, 12.

Hernandez, V. S., y Movilla, C. A. (2010). *Aplicaciones de la Teoría de Grafos en la Informática.*

Herrera, O. S. (2015). Nuevas vías en los meridianos Shu antiguos. *Revista Internacional de Acupuntura*, 12-19.

Hinz, W. A. (2012). *Graph theory of tower tasks. In Behavioural Neurology.*

JetBrains. (2016). JetBrains. Recuperado el 9 de diciembre de 2016, de https://www.jetbrains.com/pycharm/.

Larman, C. (2004). *UML y Patrones.* Recuperado el 16 de marzo de 2017, de http://www.fmonje.com/UTN/ADES%20-%20208/UML%20y%20Patrones%20%202da%20Edicion.pdf

Lluch, C. J. (2013). Grafos hamiltonianos en el diseño de viajes. En *Modelling in Science Education and Learning 6.*

Losavio, F., y Esteves, Y. (2015). Modelo del Negocio para Análisis del Dominio del Software Educativo: Un enfoque centrado en la calidad del producto. *Revista Sapiens 16*.

Lund, M. I. (2014). Diseño de Software basado en Casos de Uso. Aplicación práctica dirigida por un proceso de enseñanza-aprendizaje propuesto. *Proceedings ASSE-JAIIO*.

Marín, D. A. (16 de marzo de 2016). *Iberoamerica Divulga*. Obtenido de http://www.oei.es/historico/divulgacioncientifica/?Las-matematicas-y-su-importancia-en-nuestra-vida

Martínez, C., y Imelda, N. (2015). *Ponderación de requisitos de software usando técnicas cognitivas y orientación por objetivos.*

Martinéz, R. (2010). *PostgreSQL*. Recuperado el 9 de diciembre de 2016, de http://www.postgresql.org.es/sobre_postgresql.

REFERENCIAS

Método Experimental. (2012). Obtenido de http://fdhusacursopsicologia.webpin.com/1145949_Metodo-Experimental.html

Montero, S. I. (1 de abril de 2012). *Maestros del web.* (E. Tobar, Ed.) Recuperado el 27 de febrero de 2017, de Curso de Django para perfeccionistas: http://www.maestrosdelweb.com/guias/#guias-django

Palomino, M. Á., Gomis, V. M., y Martínez, J. P. (2006). *MaGraDa: Una Herramienta para el Tratamiento de Grafos en Matemática Discreta.* Alicante.

Park, D., Stefănescu, A., & Roşu, G. (2015). KJS: A complete formal semantics of JavaScript. *ACM SIGPLAN Notices*.

Perez, D. E., y Paumier, A. A. (2015). Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web. *Revista Latinoamericana de Ingenieria de Software*, 307.

Pressman, R. (2010). Ingeniería de Software, un enfoque práctico. Nueva York

Porto, J. P. (2015). *Definición de arista.* Obtenido de http://definicion.de/arista/

Porto, J. P., y Merino, M. (2012). *Definición de vértice.* Obtenido de http://definicion.de/arista/

Prefacio, X. V. (2000). *Parte I: El Proceso Unificado de Desarrollo de Software.*

Redondo, M. J. (2017). *Estudio con modelos de datos para la automatización en redes eléctricas inteligentes.* Obtenido de http://hdl.handle.net/10396/14506

Robledo, S., Osorio, G., y Lopez, C. (2015). Networking en pequeña empresa: una revisión bibliográfica utilizando la teoria de grafos. *Revista Vínculos,* 6.

Rocca, M. A., Valsasina, P., Meani, A., Falini, A., Comi, G., & Filippi, M. (2016). *Impaired functional integration in multiple sclerosis: a graph theory study.*

Rodríguez, A. V. (2006). *Tecnología: cuestiones generales.* Bubok Publishing.

Rodríguez, J. C. (2016). Análisis y diseño de un portal de venta de material deportivo.

Rosen, K. H. (2012). *Discrete Mathematics and Its Applications.* NewYor: McGraw-Hill Companies.

Rost, A. (2014). Scribd. Obtenido de https://es.scribd.com/document/272914355/De-que-hablamos-cuando-hablamos-de-interactividad

Saldaña, G. (2017). Nethazard. Recuperado el 18 de abril de 2017, de http://blog.nethazard.net.

REFERENCIAS

Sánchez, B., y Olmedo, J. (2015). HERRAMIENTA WEB PARA EL DIAGNÓSTICO DE VULNERABILIDADES DE SEGURIDAD Y PRUEBAS DE PENETRACIÓN.

Sánchez, F. V., y Rivero, C. A. (2015). Diseño e implementación de una estrategia de gamificacion en una plataforma virtual de educación. *Fides et Ratio-Revista de Difusión cultural y científica de la Universidad La Salle en Bolivia.*

Sebastián, F. J. (2015). *Desarrollo Web del Grupo FluIng adaptado a los nuevos estándares HTML5 y CSS3.*

Serna, E., Bedoya, i. A., y López, K. (2015). Nivel de madurez y grado de satisfacción de las herramientas libres para pruebas funcionales. *Scientia et technica*, 155-161.

Sevilla, V. d., Fernández, M. A., y Díaz., M. J. (2016). *Introducción práctica a la programación con Python.*

SILVA, I. A. (2007). *Análisis de Patrones de Software y su aplicación en un Framework de Desarrollo utilizando plataforma .net.* Obtenido de http://repositorio.espe.edu.ec/bitstream/21000/2341/1/T-ESPE-021818.pdf

Smartick. (29 de enero de 2017). Obtenido de https://www.smartick.es

Sommerville, I. (2005). Technology & Engineering. En I. Sommerville, *Technology & Engineering.*

Sommerville, I. (2006). *Software Engineering.*

Sommerville, I. (2011). Ingenieria de Software. En Somerville, *Ingenieria de Software.*

Soni, R. (2016). Introduction to Nginx Web Server. *Nginx. Apress, I*, 1-15.

Universidad Politécnica de Madrid (UPM). (2015). *Matemática Aplicada a las Tecnologías de la Información y las Comunicaciones*. Obtenido de http://www.dma.fi.upm.es/personal/gregorio/grafos/prorouting/

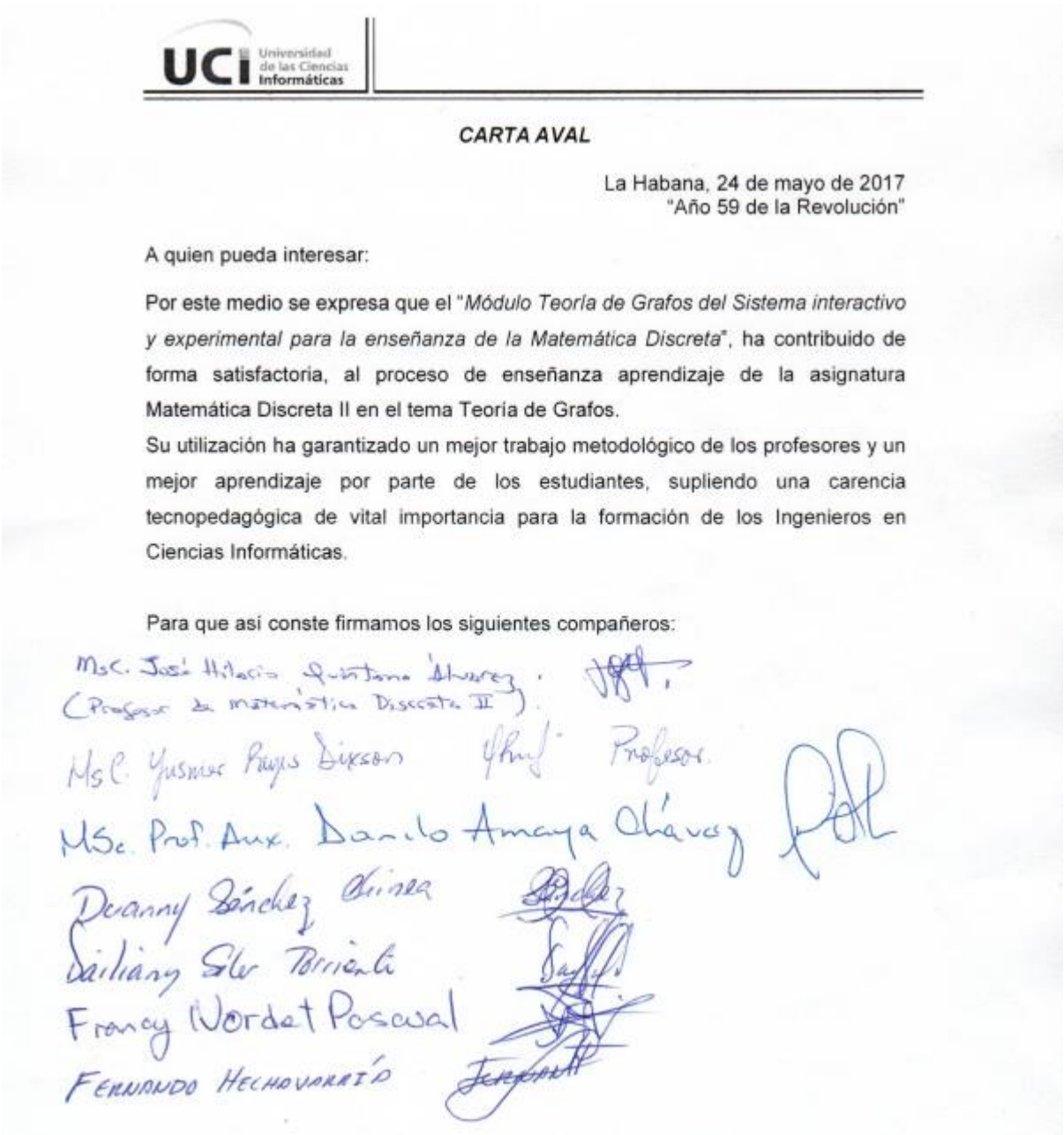
Visual Paradigm. (2013). *Visual Paradigm for uml. PARADIGM, Visual. Visual paradigm for* uml. Visual Paradigm for UML-UML tool for software application development.

Intercambios Virtuales. (28 de junio de 2017). Obtenido de http://www.intercambiosvirtuales.org/tag/wolfram-research-mathematica

Zumalacárregui, A. (2014). *Matemáticas y redes.* Madrid.

Anexos

A. Validación pedagógica



UCI Universidad de las Ciencias Informáticas

CARTA AVAL

La Habana, 24 de mayo de 2017
"Año 59 de la Revolución"

A quien pueda interesar:

Por este medio se expresa que el "*Módulo Teoría de Grafos del Sistema interactivo y experimental para la enseñanza de la Matemática Discreta*", ha contribuido de forma satisfactoria, al proceso de enseñanza aprendizaje de la asignatura Matemática Discreta II en el tema Teoría de Grafos.

Su utilización ha garantizado un mejor trabajo metodológico de los profesores y un mejor aprendizaje por parte de los estudiantes, supliendo una carencia tecnopedagógica de vital importancia para la formación de los Ingenieros en Ciencias Informáticas.

Para que así conste firmamos los siguientes compañeros:

MsC. José Hilario Quintana Alvarez. (Profesor de matemática Discreta II)

MsC. Yusmer Reyes Dixson Profesor.

MSc. Prof. Aux. Danilo Amaya Chávez

Deanny Sánchez Chinea

Dailiany Sler Borrialí

Francy Nordet Pascual

FERNANDO HECHAVARRÍA

UCI Universidad de las Ciencias Informáticas

CARTA AVAL

La Habana, 24 de mayo de 2017
"Año 59 de la Revolución"

A quien pueda interesar:

Por este medio se expresa que el "*Módulo Teoría de Grafos del Sistema interactivo y experimental para la enseñanza de la Matemática Discreta*", ha contribuido de forma satisfactoria, al proceso de enseñanza aprendizaje de la asignatura Matemática Discreta II en el tema Teoría de Grafos.

Su utilización ha garantizado un mejor trabajo metodológico de los profesores y un mejor aprendizaje por parte de los estudiantes, supliendo una carencia tecnopedagógica de vital importancia para la formación de los Ingenieros en Ciencias Informáticas.

Saludos cordiales:

MSc. Yareida Fabián Estrada
Vicedecana de Formación de la Facultad Introductoria de Ciencias Informáticas

UCI Universidad de las Ciencias Informáticas

CARTA AVAL

La Habana, 24 de mayo de 2017
"Año 59 de la Revolución"

A quien pueda interesar:

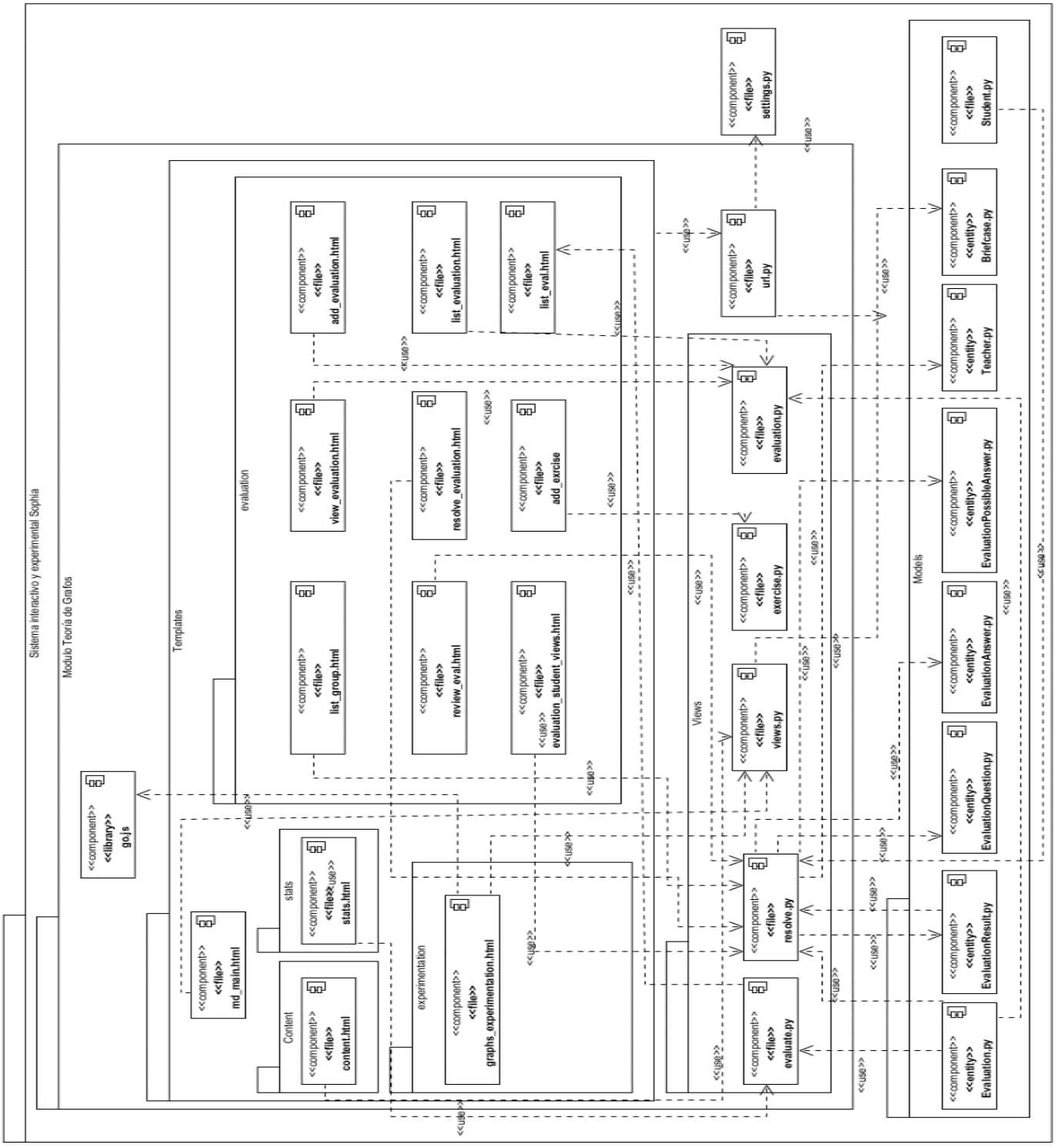
Por este medio se expresa que el "*Módulo Teoría de Grafos del Sistema interactivo y experimental para la enseñanza de la Matemática Discreta*", ha contribuido de forma satisfactoria, al proceso de enseñanza aprendizaje de la asignatura Matemática Discreta II en el tema Teoría de Grafos.

Su utilización ha garantizado un mejor trabajo metodológico de los profesores y un mejor aprendizaje por parte de los estudiantes, supliendo una carencia tecnopedagógica de vital importancia para la formación de los Ingenieros en Ciencias Informáticas.

Saludos cordiales:

Dr.C Rosa Adela González Noguera
Directora del Centro de Innovación y Calidad de la Formación

B. Diagrama de componentes



C. Diagrama de clases