Universidad de las Ciencias Informáticas

Facultad 4

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

Sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes potenciales

Autor: David Grau Rodríguez

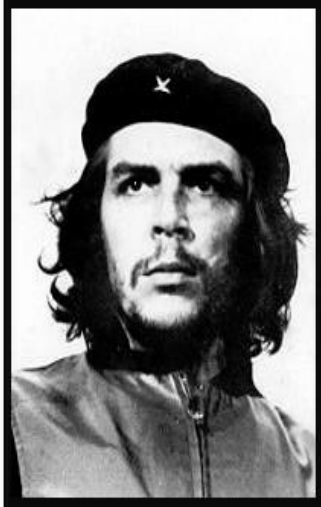
Tutores: Ing. Yasirys Terry González

Ing. Reiman Alfonzo Azcuy

Ing. Leduan Bárbaro Rosell Acosta

Ing. Naylin Corrales Sánchez

La Habana, junio, 2019



La arcilla fundamental de nuestra obra es la juventud, en ella depositamos nuestra esperanza y la preparamos para tomar de nuestras manos la bandera.

(Che Guevara)

Declaración de autoría

Declaro que soy el único autor del presente trabajo y otorgo a la Universidad de las Ciencias Informáticas los derechos primordiales del mismo, con carácter exclusivo.

Para que así conste, firmo la presente a los __ días del mes de _____ del año 2019.

_____

David Grau Rodríguez

_____ _____

Tutora: Yasirys Terry González Tutor: Ing. Reiman Alfonso Azcuy

_____

Tutor: Ing. Leduan B Rosell Acosta

Dedicatoria

A mis padres Vladimir y Caridad.

A mis hermanos Daniel, Darien y Claudia Daniela.

A mi abuela Mery y mi abuelo Wilson.

A mi novia Naylin y en especial a mi niña Khristeen.

Agradecimientos

A la Universidad de las Ciencias Informáticas por formarme y permitirme vivir estos lindos momentos.

A mis padres, que con su sacrificio, constancia y amor diario hicieron posible este sueño.

A mi hermano Daniel, por ese cariño tan inmenso que me brinda cada día y esa confianza que siempre tuvo en mí.

A mis hermanos Darien y Claudia que a veces con sus palabras inocentes, me dieron fuerzas para seguir este camino largo y difícil.

A mi novia Naylin por acompañarme durante todos estos años y darme fuerzas para no rendirme aun cuando yo pensaba que estaba todo perdido. ¡Ya podemos disfrutar juntos de nuestra bebé!

A mis tías, tíos y primos que siempre me animaron a continuar en esta travesía.

A la familia de Naylin por acogerme como un miembro más y hacerme participe de muchos lindos momentos.

A mis tutores Yasirys, Reiman y Leduan, a los cuales hoy les agradezco como tutores, pero toda la vida les estaré agradecido por ser parte indispensable de mi familia UCI.

A mis entrañables amigos de la universidad Liovan, el chino y el Mosky, con los cuales compartí casi todos los buenos y malos momentos que viví durante todos estos años.

A mi piquete del futbol.

A todas las amistades que hice a lo largo de estos años Hermes, Carrión, Luisi, Taire, Elianis, Parker, Dayron.

A todas aquellas personas que me ayudaron para que hoy este sueño se haga realidad.

Resumen

En el presente trabajo se realizó un estudio sobre los recursos educativos y su impacto en la sociedad. También se pudieron constatar las ventajas y desventajas del uso de patrones de diseño en la creación de estos recursos educativos, así como las de los agrupamientos de estos patrones en lenguajes y catálogos, con el fin de resolver problemas de mayor complejidad. Se realizó una evaluación de las aplicaciones desarrolladas en la Universidad de las Ciencias Informáticas para la recomendación de patrones de diseño. Luego del estudio realizado se concluyó que las aplicaciones existentes no satisfacían por completo las necesidades de los diseñadores de recursos educativos. Por ello, se desarrolló un sistema inteligente que, con la incorporación de métodos de agrupamiento a las soluciones previamente desarrolladas, permite sugerir la creación de posibles lenguajes de patrones. El desarrollo de esta aplicación fue guiado por la metodología AUP en su versión UCI, enmarcada en el escenario 4, que se enfoca en la descripción de historias de usuario. Se utilizaron los lenguajes de programación Java y JavaFX y el entorno de desarrollo Netbean 8.2, así como la herramienta SceneBuilder. Se validó el correcto funcionamiento de la aplicación a través de las pruebas de software, detectando y corrigiendo las no conformidades.

Palabras claves:

Patrones de diseño, lenguajes, inteligente, recomendar, agrupamiento.

Índice

Índice de ilustraciones

Índice de tablas

Introducción

Con la utilización de nuevas Tecnologías de la Información y las Comunicaciones (TICs), se ha revolucionado el proceso de enseñanza-aprendizaje. Las TICs han enriquecido y transformado la educación, brindando a la sociedad nuevas formas de educar. La utilización de las TICs disminuye los costos y aumenta la accesibilidad de la población al proceso de enseñanza. Las TICs juegan un papel fundamental en los procesos educacionales de cada país[1].

Mediante la World Wide Web (www) se puede acceder a diversas páginas web, ubicadas en disímiles partes del mundo. Estas páginas comparten conocimiento respecto a las diversas formas en que la tecnología puede facilitar el acceso universal a la educación. Con el uso de herramientas cada vez más sofisticadas y la información brindada en estos servidores, ha aumentado progresivamente la creación de Recursos Educativos(RE) [1].

Los RE son cualquier instrumento u objeto que pueda servir como recurso para que, mediante su manipulación, observación o lectura, se ofrezcan oportunidades de aprender algo, o bien con su uso se intervenga en el desarrollo de alguna función de la enseñanza. Son los medios que vehiculizan un mensaje con fines de enseñanza. Los materiales educativos presentan contenidos a través de uno o más medios. Se entiende por recurso educativo un objeto que facilita una experiencia de aprendizaje, es decir, una experiencia de cambio y enriquecimiento en algún sentido: conceptual o perceptivo, afectivo, de habilidades o actitudes[2].

Con la utilización de los recursos educativos, los distintos métodos y medios de educación han sufrido grandes cambios. Debido a la revolución educacional impulsada por los recursos educativos, estos han tenido que evolucionar a medida que pasa el tiempo, dando paso a los Recursos Educativos Digitales(RED).

Los RED son materiales compuestos por medios digitales y producidos con el fin de facilitar el desarrollo de las actividades de aprendizaje. Un material didáctico es adecuado para el aprendizaje si ayuda al aprendizaje de contenidos conceptuales, ayuda a adquirir habilidades procedimentales y ayuda a mejorar la persona en actitudes o valores[3]. Son recursos interactivos y dinámicos, ya que presenta diferentes elementos multimediales como las imágenes, sonidos, videos, animaciones, entre otros. La innovación tecnológica ha permitido tener disponible una diversidad de recursos digitales para fines de aprendizaje. Es así como en la actualidad docentes y estudiantes acceden tanto a software

educativo como a sitios Web educativos, con la finalidad de fortalecer, mejorar y contextualizar sus prácticas educativas[4].

Teniendo en cuenta la gran cantidad de recursos educativos y las diferentes utilidades para las que son creados, estos recursos se vuelven cada vez más complejos. Siendo esto un problema difícil de superar para desarrolladores inexpertos o con poco tiempo para dedicar al desarrollo de los mismos. Por tal motivo, para la creación de estos RE a sus creadores se les recomienda el empleo de patrones de diseño de recursos educativos, con el objetivo de potenciar en estos, características como mejor estructura y reusabilidad.

Se define como un patrón de diseño a una solución probada para un problema en un contexto. Cada uno documenta una solución reutilizable, encapsula el conocimiento sobre la práctica exitosa y proporciona información sobre su utilidad y sus compensaciones. Algunos patrones han sido catalogados en colecciones o bibliotecas de patrones[5-8].

Los patrones de diseño de recursos educativos se pueden almacenar en repositorios, con el objetivo de prestar servicios a los usuarios que deseen acceder a ellos. En la actualidad, existen numerosos repositorios de este tipo, algunos de ellos son Pedagogical Pattern (PPP), E-LEN, PCeL, P-REPLIKA; donde se puede encontrar patrones de diseño, que a su vez están agrupados en colecciones como lenguajes y catálogos.

En el libro *"A Pattern Language"* de Christopher Alexander está implícita la idea de que los patrones deben estar organizados por el tipo de problema que resuelven. Los patrones de diseño deben estar relacionados los unos con los otros para poder formar un lenguaje de patrones, donde cada patrón debe indicar su relación con otros patrones y con el lenguaje en sí[6].

Lo catálogos son un grupo de patrones clasificados por uno o más criterios y relacionados entre sí, los cuales pueden ser utilizados de forma conjunta o independiente. Estos criterios permiten organizar los patrones en grupos que comparten el mismo conjunto de propiedades, y dependiendo de los criterios elegidos se pueden definir esquemas de clasificación con diferentes dimensiones. Los esquemas de clasificación ponen de manifiesto las principales cualidades de los patrones y ayudan a reducir el tamaño del espacio de búsqueda[9]. Para la creación de recursos educativos es necesario el empleo de patrones de diseño, pues estos proporcionan a los diseñadores mayores probabilidades de crearlos con mejor calidad. Debido a la gran variedad de recursos educativos y a sus disímiles características, a los diseñadores se les dificulta la selección de los mismos.

En la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI) se creó en el 2015 la aplicación Sistema de Recomendación de Patrones de Diseño de Recursos Educativos Abiertos (REPREA). Esta aplicación es un sistema de recomendación de patrones basado en conocimiento, la misma utiliza información de un dominio para arribar la solución de un problema a través de una similitud de cadenas entre el problema introducido y el problema del patrón almacenado, recomienda una serie de patrones disponibles en los repositorios [10]. Los patrones recomendados son considerados posibles soluciones, pero al ser utilizados como patrones aislados, se dificulta el proceso de creación de recursos educativos complejos. Dentro de las soluciones propuestas, se encuentra la utilización de lenguajes y catálogos.

En el curso 2017-2018 se desarrolló en la Facultad 4 de la UCI la aplicación REP-TCP, la que recomienda patrones y lenguajes de patrones de diseño de recursos educativos a partir del tratamiento de agrupaciones y la calidad percibida por los usuarios. Esta aplicación solo recomienda patrones agrupados en lenguajes y en catálogos creados previamente, dificultándole a los diseñadores encontrar soluciones a nuevos problemas, para los cuales no se han definido soluciones previas en los repositorios. Por tal motivo, es necesario crear una aplicación que recomiende lenguajes potenciales de patrones de diseño, a partir del agrupamiento de patrones que den solución a problemas comunes o que los mismos sean recomendados por los usuarios para resolver problemas de características similares.

A partir de las limitaciones encontradas en REP-TCP, se origina el siguiente **problema a resolver:** ¿Cómo agrupar patrones de diseño de recursos educativos que pudieran conformar lenguajes?

Se define como **objeto de estudio**: la recomendación de contenidos digitales, siendo el **campo de acción:** el agrupamiento de patrones de diseño de recursos educativos para conformar lenguajes.

El **objetivo general** de la investigación es: desarrollar un sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes potenciales.

Como **objetivos específicos** se determinaron los siguientes:

1. Fundamentar los conceptos, características y antecedentes de los sistemas de recomendación y los algoritmos de agrupamiento mediante un estudio bibliográfico.

2. Diseñar el sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes.
3. Implementar el sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes.
4. Realizar pruebas al sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos para conformar lenguajes potenciales.

Métodos teóricos

Para la realización de esta investigación se utilizarán métodos que permiten descubrir en el objeto de estudio las relaciones esenciales y las cualidades fundamentales.

Los métodos que serán empleados son:

- **Histórico-Lógico:** este método será usado para hacer un estudio acerca del origen de los sistemas de recomendación y su evolución hasta la actualidad. Este estudio permitirá adquirir referentes teóricos e históricos para ver cómo se han ido perfeccionando a través del tiempo y sirva como base para el desarrollo del sistema como resultado de la presente investigación.
- **Analítico-Sintético:** este método será utilizado para analizar y resumir la información adquirida de bibliografías y materiales sobre el proceso de recomendación y los patrones de diseño de recursos educativos. Posibilita un mayor entendimiento acerca del funcionamiento de los sistemas de recomendación y conocer mejor sus principales características, para escoger las que tendrá el sistema a desarrollar como resultado de la presente investigación.
- **Entrevista**: la entrevista utilizada es la no estructurada, por ser la más flexible, de manera tal que se adapte a los sujetos y las condiciones del lugar donde se realiza. Se entrevista al cliente para conocer las nuevas funcionalidades que se deben incorporar al sistema.
- **Análisis documental:** este método consiste en un conjunto de operaciones encaminadas a representar un documento y su contenido bajo una forma diferente de su forma original, con la finalidad de posibilitar su recuperación. Es una operación intelectual que da lugar a un subproducto o documento secundario que actúa como intermediario o instrumento de búsqueda obligado entre el documento original y el usuario que solicita información. El calificativo de intelectual se debe a

que el documentalista debe realizar un proceso de interpretación y análisis de la información de los documentos y luego sintetizarlo.

- **Criterio de expertos:** este método consiste en la selección de un grupo de expertos, los cuales serán los encargados de valorar un grupo de aspectos seleccionados previamente por el investigador. Este método consiste en dos momentos fundamentales, inicialmente se les solicita a los expertos que hagan sus propuestas acerca de los aspectos iniciales de la investigación a evaluar. En un segundo momento se confecciona un instrumento que permita la valoración de los principales aspectos relacionados con el tema de investigación, los cuales se modelan a partir del análisis de resultados del estudio de la bibliografía y de los criterios del primer momento.

Capítulo 1. Marco teórico sobre los sistemas de recomendación y los patrones de diseño de recursos educativos

En el presente capítulo se describen conceptos fundamentales para la comprensión de las características de los recursos educativos, los patrones de diseño, los lenguajes de patrones de diseño y los sistemas de recomendación. También se hace referencia a los algoritmos de aprendizaje automático, así como las herramientas y tecnologías utilizadas para la implementación del sistema.

1.1 Recursos educativos

Los recursos educativos son materiales o herramientas con intención didáctica para apoyar el desarrollo de los procesos de enseñanza-aprendizaje. Los RE aportan información, sirven para poner en práctica lo aprendido y constituyen guías para los alumnos.

Los RE incluyen: cursos completos, materiales para cursos, módulos, libros de texto, vídeos, pruebas, software y cualquier otra herramienta que contribuya el apoyo para acceder al conocimiento [2, 5, 10].

Los RE pueden estar compuestos por [2, 5]:

- **Contenidos educativos:** programas educativos, módulos de contenido, objetos de aprendizaje, materiales multimedia (texto, sonido, video, imágenes, animaciones), exámenes, publicaciones periódicas (diarios y revistas), etc.
- **Herramientas:** software para apoyar la creación, uso y mejoramiento de contenidos educativos. Esto incluye herramientas y sistemas para: crear, registrar y organizar contenido; gestionar el aprendizaje y desarrollar comunidades de aprendizaje en línea.
- **Recursos de implementación:** licencias de propiedad intelectual que promuevan la publicación abierta de materiales, principios de diseño, adaptación, localización de contenido. Por lo general, quienes crean recursos educativos, permiten que cualquier persona use sus materiales, los modifique, los traduzca o mejore, además, que los comparta con otros. Se deben tener en cuenta que algunas licencias restringen las modificaciones (obras derivadas) o el uso comercial.

- **Enlaces externos:** observatorios y centros de información para la promoción del uso, creación y difusión de estos recursos.

Los RE cumplen las siguientes características:

- Su objetivo es apoyar el proceso de enseñanza-aprendizaje a partir de su propósito educativo.
- Según la licencia bajo la cual se encuentren, se pueden reutilizar como un todo o solamente una de sus partes.
- Son volátiles pues permiten la adaptación, mezcla y mejora.

1.2 Patrones de diseño

Christopher Alexander en 1977 definió por primera vez el término de patrón de diseño, donde propuso la idea del patrón como aquel que: “describe un problema que ocurre una y otra vez y, a continuación, describe el núcleo de la solución de ese problema, de tal manera que el usuario puede utilizar esta solución un millón de veces más, sin tener que hacerlo de la misma manera dos veces” [6].

El ministerio de educación del gobierno español en su página oficial define los patrones de diseño como [9]: Una forma literaria de documentar las mejores prácticas y lecciones aprendidas en la resolución de un problema complejo dentro de un dominio de diseño concreto, ya sea éste la arquitectura urbanística, el desarrollo de software o el diseño de cursos. Teniendo en cuenta que una de las cualidades principales de los patrones debería ser su capacidad para comunicar soluciones reutilizables a personas sin experiencia en ese dominio de diseño, un factor crítico es cómo realizar dicha descripción de tal manera que el destinatario sea capaz de comprender la motivación, la esencia y la utilidad del patrón. Es decir, que sea capaz de decidir si va a utilizar el patrón o no para resolver el problema al que se enfrenta.

Miguel Zapata aporta otro concepto más específico, donde pone de manifiesto algunos contextos donde se puede utilizar un patrón. Él plantea que los patrones son: “Estructuras de información que permiten resumir y comunicar la experiencia acumulada y la resolución de problemas, tanto en la práctica como en el diseño, en programas de enseñanza y aprendizaje a través de redes” [11].

Los patrones de diseño se pueden visualizar como una solución a un problema en un contexto dado, tres de sus aspectos fundamentales que se destacan son los siguientes:[5, 12].

- **Nombre del patrón:** Debe ser un nombre significativo y descriptivo en cuanto al problema tratado, de ser posible, formado por una sola palabra o expresión descriptiva que resuma su contenido.
- **Problema:** Descripción del problema cuya solución se propone. Suele incluir una descripción del propósito del problema que se presenta y si la solución propuesta es aplicable en cuanto a su contexto.
- **Solución:** Descripción clara de cómo alcanzar el resultado deseado. Es el equivalente a las instrucciones de aplicación del patrón. Es conveniente recoger uno o más ejemplos claros de aplicación del patrón (su contexto inicial de aplicación, el modo de aplicación, los resultados). Recoger un ejemplo claro ayuda a entender el uso y aplicación del patrón.

La utilización de patrones de diseño proporciona numerosos beneficios para la confección de un recurso educativo [12]:

- Brinda una fácil reutilización de diseños para crear recursos educativos.
- Facilita, a los diseñadores, el acceso a técnicas o diseños previos exitosos.
- Permite seleccionar diferentes alternativas de solución para un problema en específico.
- Mejora notablemente la documentación y el mantenimiento de sistemas existentes.

Actualmente, existen numerosos repositorios donde se almacenan patrones de diseño, por ejemplo, Pedagogical Pattern (PPP), E-LEN, PCeL y REPLIKA. Algunos de ellos agrupan patrones en colecciones (lenguajes y catálogos).

1.3 Catálogos y lenguajes de patrones de diseño

Los patrones de diseño de recursos educativos, atendiendo a sus características pueden ser agrupados en dos grandes grupos, los catálogos y los lenguajes. En el presente epígrafe se abordarán las principales características de dichos grupos, haciendo énfasis en las distintas formas de agrupar los patrones en los mismos.

1.3.1 Catálogos de patrones de diseño

Lo catálogos de patrones de diseño son un grupo de patrones clasificados por uno o más criterios y relacionados entre sí, los cuales pueden ser utilizados de forma conjunta o independiente. Estos criterios permiten organizar los patrones de diseño en grupos que comparten el mismo conjunto de propiedades, y dependiendo de los criterios elegidos se pueden definir esquemas de clasificación con diferentes dimensiones. Los esquemas de clasificación ponen de manifiesto las principales cualidades de los patrones y ayudan a reducir el tamaño del espacio de búsqueda[9].

La definición o selección de los criterios de organización para un determinado catálogo dependerá del objetivo de una clasificación específica, en este caso sería, ayudar a los diseñadores de recursos educativos seleccionar los patrones de diseño pertenecientes a ese catálogo que responden ante un problema de diseño al que se están enfrentando. Por esta razón, los catálogos son un conjunto de patrones que se unen respondiendo a una característica común, generalmente el tipo de problema que resuelven[5].

Los catálogos consisten en un grupo de patrones interrelacionados entre sí. Por lo general, son creados agrupando un patrones teniendo en cuenta el tipo de problema que resuelven[13].

1.3.2 Lenguajes de patrones de diseño

Un lenguaje de patrones es un método para describir buenas prácticas de diseño o patrones de organización útil dentro de un campo de experiencia. El término fue acuñado por el arquitecto Christopher Alexander y popularizado por su libro de 1977 “A Pattern Language”[6].

José María Rodríguez define un lenguaje de patrones de diseño como: un método que estructure la descripción de una serie de buenas prácticas de diseño en un área determinada conforma un lenguaje de patrón. De este modo, un lenguaje de patrón permitirá denominar de forma clara y fácil los problemas más frecuentes en un área de conocimiento o en un ámbito de trabajo determinado, así como las soluciones adecuadas para cada uno de ellos, de modo que se facilitan diferentes posibilidades de diseño de un determinado proceso”[12].

Los patrones de diseño que forman parte de un lenguaje de patrones se almacenan en un repositorio de patrones accesible a través de la web. El repositorio proporciona un índice de todos los patrones de diseño y contiene vínculos de hipermedias que permiten al usuario entender las colaboraciones entre esos patrones[14].

Por las razones antes expuestas, se puede concluir que los lenguajes de patrones son un conjunto de patrones creados para resolver por sí solos problemas específicos, pero al relacionarlos entre sí, resuelven problemas mayores.

1.4 Sistema de recomendación precedente

Con el objetivo de facilitar el trabajo de los desarrolladores de recursos educativos en la UCI, en el año 2018 se desarrolló el Sistema para recomendar patrones de diseño de recursos educativos basado en el tratamiento de agrupaciones y la calidad percibida por los usuarios. Este sistema recomienda a los usuarios los patrones de diseño independientes, los incluidos en catálogos y los lenguajes de patrones que podrían constituir soluciones. Las interacciones del usuario con el sistema y el funcionamiento general de este último responden a los siguientes pasos[5]:

1. El usuario introduce el problema de diseño de recursos educativos para el que necesita emplear patones
2. El sistema calcula la medida de semejanza entre los textos de las descripciones de los problemas: el introducido por el usuario y el correspondiente a los patrones, catálogos y lenguajes almacenados.
3. El sistema selecciona aquellos patrones de diseño y lenguajes cuya medida de semejanza arrojó un valor igual o superior al umbral definido.
4. El sistema realiza un ordenamiento de los patrones seleccionados en el paso anterior, atendiendo a los parámetros de calidad percibida por los usuarios, que son: la cantidad de descargas, la cantidad de ocasiones en que se ha recomendado y el promedio de las evaluaciones recibidas por otros usuarios.

El Sistema para recomendar patrones de diseño de recursos educativos basado en el tratamiento de agrupaciones y la calidad percibida por los usuarios responde a la concepción de un sistema de recomendación híbrido, que incluye la recomendación colaborativa y la recomendación basada en conocimiento, tal como se evidencia en la

propia descripción de su funcionamiento. La solución que se propondrá como resultado del presente trabajo contará con un alto grado de independencia del actual proceso de recomendación, por lo que será necesario analizar cuál o cuáles de los tipos de recomendación se debe(n) aplicar.

1.4.1 Calidad percibida por los usuarios

Para realizar la recomendación de patrones de diseño de recursos educativos el sistema desarrollado gestiona la información existente sobre los diferentes elementos básicos que se manejan en su composición: las preferencias de los usuarios, los catálogos, los lenguajes, los patrones de diseño y los repositorios de patrones [5, 15, 16].

Los elementos a tener en cuenta para calcular la calidad percibida por los usuarios son[5]:

- **Cantidad de descargas:** con este parámetro se obtendrá la cantidad de veces que los usuarios han descargado determinado patrón, lenguaje o patrones pertenecientes a catálogos.
- **Cantidad de recomendaciones:** con este parámetro se obtendrá la cantidad de veces que los usuarios han recomendado con anterioridad determinado patrón, lenguaje o patrones pertenecientes a catálogos.
- **Promedio de evaluaciones:** con este parámetro se obtendrá el promedio de evaluaciones acerca del criterio que tengan los usuarios de determinado patrón, lenguaje o patrones pertenecientes a catálogos.

La información de estos parámetros es obtenida a través de los resultados de la interacción de los usuarios con determinado patrón de diseño mediante un indicador de interacción. El cálculo de este indicador se realiza para aquellos lenguajes, patrones pertenecientes a un catálogo y patrones independientes seleccionados en la fase anterior y toma valores entre 0 y 1, se calcula de la siguiente forma[5, 15, 16]:

$$INT(p) = \sum_{i=1}^{v} \frac{0{,}5(\frac{E_i}{5}) + 0{,}3R_i + 0{,}2D_i}{v}$$

Ilustración 1. Ecuación para determinar la calidad percibida por los usuarios.

Donde:

- *v* es la cantidad de veces que el patrón ha sido visualizado.
- E_i es la evaluación otorgada al patrón por el usuario en la visualización *i*, toma valor entero entre 1 y 5.
- R_i y D_i toman valor 1 si el usuario ha recomendado y descargado, respectivamente, el patrón en la visualización *i* y valor 0 en caso contrario.

Luego del cálculo de la calidad percibida, esta es utilizada para establecer un orden de prioridad en los resultados que serán mostrados como posibles soluciones.

Con el estudio del Sistema para recomendar patrones de diseño de recursos educativos basado en el tratamiento de agrupaciones y la calidad percibida por los usuarios, se pudo constatar que es un sistema que responde a necesidades reales de los diseñadores de recursos educativos, por lo que es necesario que se mantenga funcional. Teniendo en cuenta la calidad que aporta a las recomendaciones del sistema, el método para ordenar, basado en la calidad percibida por los usuarios, también será utilizado para ordenar los resultados obtenidos por el algoritmo de agrupamiento que se utilizará en la solución resultante del presente trabajo.

1.5 Sistemas inteligentes

Se puede definir un sistema inteligente como un programa de computación que cuenta con características y comportamientos similares a los de la inteligencia humana o animal, es decir, que cuenta con la capacidad de decidir por sí mismo qué acciones realizará para alcanzar sus objetivos basándose en sus percepciones, conocimientos y experiencias acumuladas[17, 18].

Para que un sistema pueda ser considerado inteligente, debe presentar diversas funcionalidades que incluyan:

- **Inteligencia:** es el nivel del sistema en lograr sus objetivos.
- **Objetivo:** Un objetivo es una cierta situación que el sistema inteligente quiere lograr. Normalmente hay muchos niveles de objetivos, puede haber un objetivo principal y muchos sub-objetivos.

- **Capacidad sensorial:** Un sentido es la parte del sistema que puede recibir comunicaciones del entorno. Se necesitan los sentidos para que el sistema inteligente puede conocer su entorno y actuar interactivamente.
- **Conceptualización:** Un concepto es el elemento básico del pensamiento. Es el almacenamiento físico o material de información. Todos los conceptos de la memoria están interrelacionados en red. La capacidad de conceptualizar implica el desarrollo de niveles de abstracción.
- **Reglas de actuación:** Una regla de actuación es el resultado de una experiencia o el resultado de interpretar la propia memoria. Relaciona situación y consecuencias de la acción.
- **Memoria:** La memoria es un almacenaje físico de conceptos y reglas de actuación. Esto incluye la experiencia del sistema.
- **Aprendizaje:** El aprendizaje es probablemente la capacidad más importante de un sistema inteligente. El sistema aprende conceptos a partir de la información recibida de los sentidos. Aprende reglas de actuación a base de su experiencia. La actuación, a veces hecha al azar, se almacena con su valor. Una regla de actuación aumenta en valor si permitió el logro de un objetivo. El aprendizaje incluye la fijación de conceptos abstractos, a base de ejemplos concretos y la creación de conceptos compuestos que contienen los conceptos de partes de un objeto. El aprendizaje también es la capacidad de detectar relaciones (patrones) entre la parte "situación" y la parte "situación futura" de una regla de actuación.

Con el estudio realizado sobre las características que deben estar presente en un sistema para considerarlo inteligente. Se pudo constatar que el sistema desarrollado es un sistema inteligente, pues el mismo contiene en sí, las características necesarias para considerarlo como tal.

1.6 Sistemas de recomendación

Con la aparición de la web y su posterior desarrollo, los sistemas de recomendación surgen como forma de que los usuarios puedan conocer qué productos son los que más les interesan dentro de la gran variedad en la que pueden escoger. Por tanto, estos sistemas ayudan a encontrar productos que cubran sus necesidades y se encuentren dentro de sus

preferencias, haciendo más fácil la tarea de clasificar y adquirir productos a través de las nuevas tecnologías[19].

Los sistemas de recomendación tienen como principal objetivo brindar a los usuarios resultados de búsqueda cercanos o adaptados a sus necesidades, realizando predicciones de sus preferencias y entregando aquellos *ítems* que podrían acercarse más a lo esperado. Son los aliados de la personalización de sistemas computacionales, por su capacidad de identificar preferencias y sugerir *ítems* relevantes para cada usuario; para ello se necesita de perfiles que almacenen la información y las preferencias de cada usuario[20].

Un sistema de recomendación es aquel sistema que busca predecir la importancia o preferencia que un usuario puede dar a un determinado *ítem* o grupo de ítems. Esta capacidad se ha visto muy importante en aplicaciones de comercio electrónico, pero también en otros como sistemas expertos, de educación, citas en líneas y redes sociales[21].

Aunque existen diversos tipos de sistemas de recomendación, todos necesitan una gran cantidad de información sobre los usuarios para poder realizar recomendaciones de calidad. Teniendo en cuenta la importancia que los usuarios le conceden a los sistemas de recomendación, es necesario elegir correctamente las entradas necesarias para los algoritmos de recomendación utilizados. Dichas entradas son el pilar fundamental para una buena recomendación. Otro de los pilares fundamentales en una recomendación es conocer bien las características de los usuarios a los cuales va dirigida la recomendación, asegurando así que esta esté dirigida a las necesidades del grupo.

1.6.1 Tipos de sistemas de recomendación

Los sistemas de recomendación utilizan técnicas basadas en la información que pueden obtener del conocimiento de los productos, de las valoraciones de los usuarios, de la interacción del usuario con el recomendador o de la interacción del usuario con los productos. En base a esto, se realiza una clasificación en diferentes tipos de recomendación (esta es la clasificación más extendida)[19]:

- Los sistemas de recomendación basados en **filtrado colaborativo**. Se basan en las valoraciones que han dado los usuarios sobre los productos.
- Los sistemas de recomendación **basados en contenidos**. Son los que tienen en cuenta la descripción de los productos para hacer la recomendación.

- Los sistemas de recomendación **basados en conocimiento**. Son aquellos que utilizan todo el conocimiento que se puede obtener de los usuarios y productos. Se basa en inferencias sobre las necesidades y preferencias de los usuarios.
- Los sistemas de recomendación **híbridos**. Se trata de sistemas que combinan técnicas de los anteriores recomendadores para intentar cubrir sus diferencias.

Los repositorios donde se almacenan recursos educativos, se caracterizan por no almacenar perfiles de usuarios, los datos más comunes que se pueden obtener son los correspondientes a la interacción del usuario con los patrones. Por esta razón, en la propuesta de solución, se utilizará como tipo de sistema de recomendación, el basado en el filtrado colaborativo.

1.7 Análisis de soluciones similares

Con el objetivo de realizar un estudio de las soluciones desarrolladas con antelación para la creación sistemas similares, fueron analizados los siguientes sistemas:

- Se analizó el trabajo “Estructuración de la base de conocimiento en un sistema basado en casos utilizando algoritmos conceptuales”, publicado en la Revista Cubana de Ciencias Informáticas que sugiere una propuesta de procedimientos a seguir para estructurar jerárquicamente la base de conocimiento en un sistema basado en casos, utilizando los algoritmos conceptuales del reconocimiento lógico combinatorio de patrones que favorece el acceso y recuperación de los casos semejantes para dar solución a problemas caracterizados por datos mezclados e incompletos. Para comprobar la eficiencia del modelo de la base de casos propuesto, este debe compararse con una estructuración plana de la base de casos. Es por ello, que en el experimento se incluye el algoritmo K-means, además se evalúa el comportamiento de otro enfoque representativo de las organizaciones jerárquicas conceptuales a través del algoritmo Cobweb y para completar el análisis se incluyen resultados respecto a algoritmos que forman árboles de decisión como C4.5. Para realizar las pruebas a estos algoritmos se utilizaron 15 conjuntos de datos de reconocimiento internacional disponibles en el repositorio para aprendizaje automatizado de la Universidad de Irvine, California. Dichos conjuntos están conformados por datos como las características de diversos autos, tipos de enfermedades dermatológicas, entre otros [27].

- Se analizó el trabajo “Sistema de recomendación híbrido para la predicción de calificaciones en Yelp.com” publicado en la Universidad Politécnica de Madrid. En dicho trabajo se propone una solución para un problema de recomendación planteado por Yelp.com. Para dar solución a este problema, se realiza un estudio de los diferentes sistemas de recomendación, a partir de soluciones similares encontradas. Se realiza un estudio de diferentes algoritmos utilizados para el modelado de datos, tales como C4.5, Regresión lineal múltiple, algoritmo colaborativo usuario a usuario, algoritmo colaborativo *ítems* a *ítems* y factorización de matrices. Los algoritmos estudiados fueron implementados utilizando la librería Weka. Se aplican diferentes técnicas de minería de datos, se analizan ventajas y desventajas de cada una de estas y finalmente se propone un modelo híbrido que ayuda a predecir estas calificaciones con la confianza y exactitud deseados. Los datos utilizados para realizar la comparación entre los algoritmos fueron extraídos de una base de conocimiento brindada por la empresa Yelp[21].

- Se analizó el trabajo “Un enfoque híbrido para la calificación de imágenes de resonancia magnéticas del cerebro” publicado en la Universidad Nacional de San Agustín. En dicha publicación se realiza un estudio de las distintas redes neuronales artificiales y sus diversas aplicaciones. Luego del estudio realizado se plantea la utilización de un sistema de recomendación híbrido formado por una red neuronal y un algoritmo genético para detectar en una imagen de resonancia magnética si tiene o no una anomalía[28].

- Se analizó el trabajo “Sistema de recomendación web basado en la actividad de los usuarios de la Universidad Nacional de Colombia” publicado en la facultad de ingeniería de la Universidad Nacional de Colombia. En dicho trabajo se realiza un estudio de los sistemas de recomendación existentes, para posteriormente seleccionar como tipo de recomendación a utilizar la recomendación híbrida. También se realiza un estudio del algoritmo K-means, analizando los resultados obtenidos con el algoritmo principal y las modificaciones conocidas del mismo. Por otra parte, también fue analizado el algoritmo Simple Aglomerativo, el cual produce secuencias anidadas de grupos, donde en el nivel más bajo de la jerarquía se encuentran los objetivos individuales y a medida que esta avanza se forman los *clusters* hasta llegar a la sima. Otro algoritmo analizado fue el Chameleon, el cual involucra los conceptos de cercanía relativa y de interconectividad relativa entre

los grupos. Para realizar la comparación entre estos algoritmos se utilizó como datos de entrada, la interacción de los usuarios de la universidad con artículos en la web; teniendo como objetivo realizar una recomendación de posibles artículos de interés a visitar por estos usuarios[29].

- Se analizó el trabajo “Métodos de agrupamiento en el cálculo de distancias entre usuarios de recomendación basado en algoritmos de filtrado colaborativo”. En este trabajo se realizó un estudio de los tipos de sistemas de recomendación existentes, los tipos de algoritmos utilizados para la recomendación, teniendo en cuenta las ventajas y desventajas de cada uno. El sistema de recomendación escogido fue el sistema de recomendación colaborativo. Como parte del desarrollo de este trabajo, se realizó un estudio de algunos algoritmos de agrupamiento tales como, el algoritmo K-means y algunas de sus modificaciones conocidas. También como parte del estudio de los algoritmos de agrupamiento se analizó el algoritmo EM. Los datos de entrada utilizados para la comparación entre estos algoritmos fueron extraídos de una base de datos local perteneciente a la institución, la cual contiene información de ochenta mil usuarios almacenados[30].

- Se analizó el trabajo “Un sistema de recomendación basado en perfiles generados por agrupamiento y asociaciones”. En este trabajo se realiza un estudio de los tipos de recomendación existentes y se analiza la utilización de los mismos en sistemas como Amazon y Netflix. También se realiza un estudio de los tipos de agrupamiento existentes, lo cual sirve para arribar a la conclusión que el mejor algoritmo a utilizar para este sistema es el algoritmo K-means. Los datos utilizados para la comparación de estos algoritmos fueron extraídos de una base de datos local, la cual almacena información sobre las preferencias de los usuarios al visitar diversos sitios de ocio[31].

Tabla 1. Cuadro comparativo de las soluciones similares estudiadas.

Sistemas analizados	**Tipo de recomendación que utilizan**	**Algoritmos de agrupamiento estudiados**	**Algoritmo de agrupamiento que utilizan**
Estructuración de la base de conocimiento	No realiza recomendación	K-means,	K-means

en un sistema basado en casos utilizando algoritmos conceptuales		K-MSN, C4.5 COBWEB	
Sistema de recomendación híbrido para la predicción de calificaciones en Yelp.com	Híbrida(filtrado colaborativo, baso en contenido)	C4.5, Regresión lineal múltiple, Algoritmo colaborativo usuario a usuario, Algoritmo colaborativo *ítems* a *ítems* Factorización de matrices	C4.5, Regresión lineal múltiple, Algoritmo colaborativo usuario a usuario, Algoritmo colaborativo ítems a ítems Factorización de matrices
Un enfoque híbrido para la calificación de imágenes de resonancia magnéticas del cerebro	Híbrida(filtrado colaborativo, baso en contenido, basado en conocimiento)	Redes neuronales	Redes neuronales
Sistema de recomendación web basado en la actividad de los usuarios de la Universidad Nacional de Colombia	Híbrida(filtrado colaborativo, baso en contenido)	K-means, Simple Aglomerativo, Chameleon	K-means
Métodos de agrupamiento en el	Filtrado	K-means	K-means

cálculo de distancias entre usuarios de recomendación basado en algoritmos de filtrado colaborativo	colaborativo	EM	
Un sistema de recomendación basado en perfiles generados por agrupamiento y asociaciones	Híbrida(filtrado colaborativo, baso en contenido, basado en conocimiento)	K-means	K-means

Con el análisis de los resultados de estudios previos realizados por otros autores y de sistemas similares, se identificaron las potencialidades y desventajas de cada uno de estos sistemas y de los algoritmos de agrupamiento que constituyen sus núcleos. Teniendo en cuenta los elementos expuestos en este epígrafe, se concluye que:

- Los sistemas analizados no constituyen una solución a la problemática planteada en la presente investigación, porque los mismos están diseñados para resolver problemas específicos de las instituciones para las cuales fueron desarrollados.
- Se pudo constatar que la recomendación basada en filtrado colaborativo es una solución adecuada a implementar, debido a su extendida utilización en sistemas previamente desarrollados.
- Los algoritmos más utilizados para realizar el agrupamiento, después de realizarse estudios comparativos son: K-means, K-Mdoid, EM y redes neuronales. Por tal motivo, es necesario realizar un estudio de los algoritmos de agrupamiento utilizados en los sistemas analizados, con el objetivo que conocer cuál es el más adecuado para la implementación de este sistema.

1.8 Algoritmos de agrupamiento

A partir del estudio realizado sobre los algoritmos de agrupamiento más utilizados en los últimos tiempos, fue posible conocer el nivel de eficiencia de cada uno de ellos teniendo en cuenta la cantidad de datos a procesar. Otra de las condiciones valoradas para la comparación de los algoritmos fue el tiempo de ejecución de cada uno atendiendo a la cantidad de datos procesados. Dicho estudio arribó a las siguientes conclusiones:

- El algoritmo K-means es un algoritmo simple, efectivo para pequeñas y medianas cantidades de datos. Utiliza el promedio para representar los centros de los clusters. Permite especificar de forma inicial la cantidad de clusters o grupos deseados. La descripción del algoritmo básico es el siguiente, primero se eligen K centroides iniciales, donde K es un parámetro especificado por el usuario y corresponde al número de clusters deseados. Cada punto es asignado a su centroide más cercano y cada colección de puntos asignado a un centroide representa un cluster. El centroide de cada cluster se actualiza basado en la asignación de puntos al cluster. Se repiten los pasos de asignación y actualización hasta que los puntos dentro del cluster no cambien, o equivalentemente, hasta que los centroides dejen de cambiar. El algoritmo en cuanto a tiempo de ejecución funciona mejor, para menor cantidad de datos a evaluar[32-36].

- El algoritmo K-Medoids es mejor que k-means en cuanto a tiempo de ejecución. A diferencia del K-means, no es sensible a valores atípico pues en lugar de tomar el valor medio de los objetos en un clúster como un punto de referencia, se puede usar un Medoid, que es el objeto ubicado más centralmente en un clúster. Para el algoritmo base la complejidad de implementación es media en comparación con relación a los demás algoritmos estudiados. El tiempo de ejecución mejora cuando la cantidad de datos a evaluar es media[35, 37].

- EM pertenece a una familia de modelos que se conocen como Finite Mixture Models, los cuales se pueden utilizar para segmentar conjuntos de datos. El algoritmo obtiene muy buenos resultados cuando la base de datos está formada por nodos gaussianos, incluso cuando hay solapamiento entre sus datos. El algoritmo presenta problemas cuando los nodos de su base de datos tienen poca densidad. Teniendo en cuenta el algoritmo base posee una complejidad de implementación media con respecto a los demás algoritmos estudiados. [36, 38].

- Los algoritmos de redes neuronales artificiales (RNA) poseen gran robustez y tolerante a fallos, permite un buen comportamiento frente a cambios en el entorno o frente a datos de entrada incompletos o ruidosos. Las RNA solo pueden implementar funciones linealmente separables, aunque han surgido estructuras más complejas que permiten modelar algunos problemas no lineales. Con relación a los demás algoritmos estudiados su complejidad de implementación es alta y se necesitan grandes prestaciones de hardware para obtener buenos resultados al analizar grandes cantidades de datos. El tiempo de ejecución de una RNA en comparación con los demás algoritmos estudiados es alto[39-41].

A continuación, se muestra una tabla comparativa con las características de cada uno de los algoritmos estudiados.

Las entradas utilizadas en la siguiente tabla para realizar la comparación de los algoritmos, surgen a partir de las características que más resaltan en el estudio bibliográfico realizado.

Tabla 2. Cuadro comparativo entre algoritmos de agrupamiento

Algoritmos	Complejidad	Efectividad/cantidad de datos	Tiempo de ejecución/cantidad de datos
K-means	Baja	Pequeña y mediana	Bajo, Medio
K-Mdoid	Media	Pequeña y mediana	Medio, Bajo
EM	Media	Pequeña y mediana	Medio
Redes neuronales	Alta	Pequeña, mediana y alta	Alto

Luego del estudio realizado sobre algunos de los algoritmos de agrupamiento más utilizados en la actualidad, se pudo concluir que: el más óptimo para la implementación este sistema es el K-means, teniendo en cuenta que es el algoritmo que posee menor complejidad de implementación, brinda el menor tiempo de ejecución y efectividad para pequeñas cantidades de datos.

1.9 Metodología, herramientas y tecnologías a utilizar

Para darle cumplimiento al objetivo planteado en el presente trabajo investigativo es necesario explicar de forma detallada la metodología, herramientas y tecnologías a utilizar. Dichas metodologías y herramientas fueron seleccionadas teniendo en cuenta las utilizas en el sistema que antecede esta investigación.

1.9.1 Metodología de desarrollo de software

Una metodología es un conjunto de filosofías, etapas, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información[42].

Se determinó guiar el proceso de desarrollo del sistema a través de la metodología Proceso Unificado Ágil (AUP) en su variante UCI. Teniendo en cuenta que esta es la metodología utilizada por el Grupo de Tecnologías para el Apoyo a la Educación (GITAE) al cual tributa el sistema desarrollado. Con el uso de esta metodología se pretende que los artefactos que se generan respondan a las disposiciones establecidas para el desarrollo de proyectos en la UCI. Los sistemas desarrollados para el propio entorno donde se implementará la solución utilizan esta metodología y, por ello, mantenerla garantiza uniformidad en el proceso y en la documentación generada. Además, es una metodología híbrida, es decir, que no llega a ser robusta ni ágil, y al estar en un punto intermedio permite generar una documentación adecuada, posibilitando hacer un sistema tan específico como deseen el usuario y el desarrollador. Esta metodología propone cuatro escenarios diferentes, garantizando que se aplique satisfactoriamente a cualquier tipo de proyecto, tanto aquellos que requieren un negocio bien definido como a los que no.

Para modelar el sistema se utilizó el escenario cuatro de esta metodología, el cual define que su aplicación se debe realizar en proyectos que hayan evaluado el negocio a informatizar y como resultado se obtenga un negocio muy bien definido donde el cliente estará siempre acompañando al equipo de desarrollo. Este escenario propone elaborar Historias de Usuario (HU) en las cuales se debe establecer conversaciones acerca de las necesidades de los clientes. Además, se recomienda en proyectos no muy extensos, pues una HU no debe poseer demasiada información[43].

A continuación, se describen los objetivos de las fases definidas[43]:

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación.
- **Cierre:** En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.9.2 Herramientas y tecnologías

Lenguaje de programación: Java

Teniendo en cuenta las herramientas utilizadas en el desarrollo del sistema anterior, se decidió utilizar como lenguaje de programación Java, Javafx para la implementación de la interfaz visual. Como entorno de desarrollo se utilizó Netbeans en su versión y como gestor de bases de datos PostgreSQL [5, 10, 44, 45].

Características del lenguaje de programación Java[45]:

- Lenguaje de programación orientado a objetos.
- Permite operar de forma independiente de la plataforma y del sistema operativo que se esté utilizando.
- Los programas desarrollados en Java se compilan son portables, gracias a un lenguaje intermedio, denominado Bytecode.
- Utilizando Bytecode las aplicaciones desarrolladas en Java pueden ejecutarse en sistemas operativos como Windows, Linux, iOs o Android.

Entorno de desarrollo: NetBeans

Es una herramienta de desarrollo integrado (IDE) modular basado en estándares, escrito en el lenguaje de programación Java. Está diseñado para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Es una plataforma de código abierto, que se puede usar como un marco genérico para construir cualquier tipo de aplicación. Es compatible con las siguientes tecnologías[46]:

Tecnologías soportadas:

- Java EE 7, Java EE 6 y Java EE 5
- JavaFX 2.2.x y 8

Como requerimiento del sistema NetBeans IDE se ejecuta en sistemas operativos compatibles con Java VM (máquina virtual) y se ha probado en los sistemas operativos que presentan a continuación[46]:

- Microsoft Windows Vista SP1/Windows 7 Professional (o versiones superiores):

  Procesador: 800MHz Intel Pentium III o equivalente

  Memoria RAM: 512 MB

  Disco duro: 750 MB libre en disco

Gestor de base de datos: PostgreSQL

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto práctica como eficiente.

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más utilizado del mercado [47].

Algunas características de PostgreSQL[47]:

- Integridad referencial.
- Replicación asincrónica/sincrónica / Streaming replication-Hot Standby.

- Múltiples métodos de autentificación.
- Acceso encriptado.
- Basta documentación disponible en su página oficial y foros online.
- Disponible para Linux y UNIX y Windows 32/64bit.

Algunas de las ventajas de PostgreSQL:

- Fácil de Administrar.
- Su sintaxis SQL es estándar y fácil de aprender.
- Multiplataforma.
- Capacidades de replicación de datos.
- Soporte empresarial disponible.

Algunas de las desventajas de PostgreSQL:

- En comparación con MySQL es más lento en inserciones y actualizaciones, ya que cuenta con cabeceras de intersección que no tiene MySQL.
- Consume más recursos que MySQL.

Visual Paradigm

Visual Paradigm es una herramienta UML. Está diseñada para una amplia gama de usuarios, incluidos ingenieros de software, analistas de sistemas, analistas de negocios y arquitectos de sistemas, o para cualquier persona que esté interesada en crear sistemas de software a gran escala de manera confiable utilizando un enfoque orientado a objetos. Además, es compatible con los últimos estándares de notación UML. Está desarrollado para eliminar la complejidad, mejorar la productividad y comprimir los plazos de desarrollo de software de los clientes[48].

Ofrece una serie de facilidades para generar informes que permiten documentar el proyecto[49]:

- Generación de informes en PDF (no en la versión gratuita).
- Generación de informes en HTML (no en la versión gratuita).
- Generación de informes en Word (no en la versión gratuita).

- Publicando el proyecto.
- Escribiendo un informe.
- Ordenando los elementos de un informe.

Ofrece una serie de facilidades para Importar/Exportar modelos en diferentes formatos[49]:

- XMI.
- XML.
- Casos de Uso a/desde Word.
- Importar desde Rose, Erwin.

JavaFX

JavaFX amplía la potencia de Java permitiendo a los desarrolladores utilizar cualquier biblioteca de Java en aplicaciones JavaFX. De esta forma, los desarrolladores pueden ampliar sus capacidades en Java y utilizar la tecnología de presentación que JavaFX proporciona para crear atractivo visual. Está disponible para las plataformas Windows, Mac OS X y Linux. Posee sus propio estilo de codificación.[50].

Características principales de JavaFX[50]:

- Permite a los desarrolladores integrar gráficos vectoriales, animación, sonido y activos web de vídeo en una aplicación interactiva, completa y atractiva.
- Amplía la tecnología Java permitiendo el uso de cualquier biblioteca de Java en una aplicación JavaFX.
- Permite mantener un eficaz flujo de trabajo entre diseñador y desarrollador en el que los diseñadores pueden trabajar en las herramientas que deseen mientras colaboran con los desarrolladores.

Teniendo en cuenta que JavaFX posee su propio estilo de codificación, será necesario adaptar las funcionalidades existentes en la aplicación anterior para hacerlas compatibles con la nueva interfaz que se desarrollará.

Herramienta SceneBuilder

Scene Builder es una herramienta que facilita el trabajo con JavaFX, brindando la posibilidad de crear y modificar interfaces sin necesidad de codificarla[51].

A continuación, se muestran algunas de las principales características de la herramienta[51, 52]:

- La aplicación permite desarrollar interfaces para aplicaciones móviles y de escritorio.
- El diseño de la interfaz de usuario de arrastrar y soltar permite una iteración rápida. La separación de los archivos de diseño y lógica permite a los miembros del equipo enfocarse rápida y fácilmente en su capa específica de desarrollo de aplicaciones.
- Es gratuito y de código abierto. Cuenta con una amplia comunidad de respaldo, impulsada principalmente por una empresa conocida como Gluon. Se encuentran disponibles ofertas de soporte comercial, que incluyen capacitación y servicios de consultoría personalizados.
- Está licenciado bajo la licencia BSD.

1.10 Formato XML

El formato XML es el formato utilizado en el grupo de investigación GITAE para importar y exportar la información generada en las aplicaciones que utilizan en dicho grupo. La utilización de este formato es de gran ayuda para la estandarización de la información generada.

XML es un lenguaje de etiquetas, y la forma de modelar un proceso con este lenguaje consiste en utilizar etiquetas predefinidas que representen los diferentes elementos que pueden aparecer en la definición de un proceso, como: <actividad>, <participante>, <condición>, <estado>, <transición>, <paralelismo>, <recurso>, <secuencia>. Para que todos los modelos de procesos se basen en la misma lista de etiquetas, se debe crear lo que se conoce como una definición de estructura o esquema, que es un fichero con extensión ".xsd" al que deben hacer referencia todos los documentos XML que representen modelos de procesos basados en las mismas etiquetas[53, 54].

1.11 Conclusiones del capítulo

El análisis de las características de los recursos educativos, los patrones de diseño y los lenguajes se estableció la base conceptual para el desarrollo de la solución. A partir del estudio de los sistemas de recomendación, se decidió utilizar como tipo de recomendación

el basado en filtrado colaborativo. Teniendo en cuenta el estudio realizado sobre los algoritmos de agrupamiento, se determinó utilizar el algoritmo K-means. Se definieron como tecnologías a utilizar el lenguaje de programación Java y JavaFX, la herramienta de modelado Visual Paradigm y como gestor de bases de datos PostgreSQL.

Capítulo 2. Análisis y diseño

En el desarrollo de este capítulo se reflejan las características del sistema a desarrollar. Se utiliza el diagrama de proceso de negocio para detallar el sistema desarrollado. Se diseña y describe la propuesta de solución. Se refleja la captura de requisitos funcionales y no funcionales, las historias de usuario y los artefactos generados en las fases de ejecución.

2.1 Propuesta de solución

Para dar solución a la situación problémica planteada en la presente investigación se implementa el sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes. El sistema cuenta con las características de recomendación de patrones de diseño pertenecientes a catálogos, lenguajes de patrones de diseño o patrones de diseño independientes, heredadas del Sistema para recomendar patrones de diseño de recursos educativos basado en el tratamiento de agrupaciones y la calidad percibida por los usuarios. Además, el sistema brinda al usuario la posibilidad de consultar la recomendación de nuevos lenguajes de patrones de diseño de recursos educativos, sugeridos a partir de la utilización de técnicas de agrupamiento, aplicadas a la información almacenada en la base de datos del sistema.

Cuando el usuario selecciona la opción de “Agrupar patrones”, el sistema analiza mediante técnicas de aprendizaje automático los patrones independientes que han sido utilizados para resolver problemas similares. Para la implementación de estas técnicas de aprendizaje automático fue utilizado el algoritmo K-means de la librería Weka. Las entradas utilizadas para la implementación del algoritmo son las siguiente:

- El número de *clusters* que realizará el algoritmo.
- La cantidad de iteraciones que realizará el mismo.
- La lista de patrones independientes y sus datos de interacción, los cuales se encuentran almacenados en la base de datos local.

La cantidad de *clusters* utilizados en el algoritmo se calcula automáticamente mediante un algoritmo de inteligencia artificial utilizado en sistemas de agrupamientos precedentes al sistema desarrollado. El algoritmo creará los *clusters* teniendo en cuenta la fecha en que los patrones han sido consultados por los usuarios. Luego del agrupamiento desarrollado por el algoritmo K-means, el sistema realiza un ordenamiento de los resultados obtenidos, a

partir, del índice de interacción de los usuarios con los patrones almacenados, utilizando para el cálculo de este índice la calidad percibida por los usuarios.

Para el cálculo de esta calidad percibida son tenidas en cuenta características como la cantidad de descargas realizadas a cada patrón, la cantidad de visualizaciones y las evaluaciones realizadas al patrón. Posteriormente el sistema brinda la posibilidad al usuario de exportar el resultado obtenido en formato XML, para su posterior revisión y utilización, en caso de que el mismo lo considere necesario. La estructura del archivo XML utilizada es la misma estructura estudiada en el epígrafe 1.10.

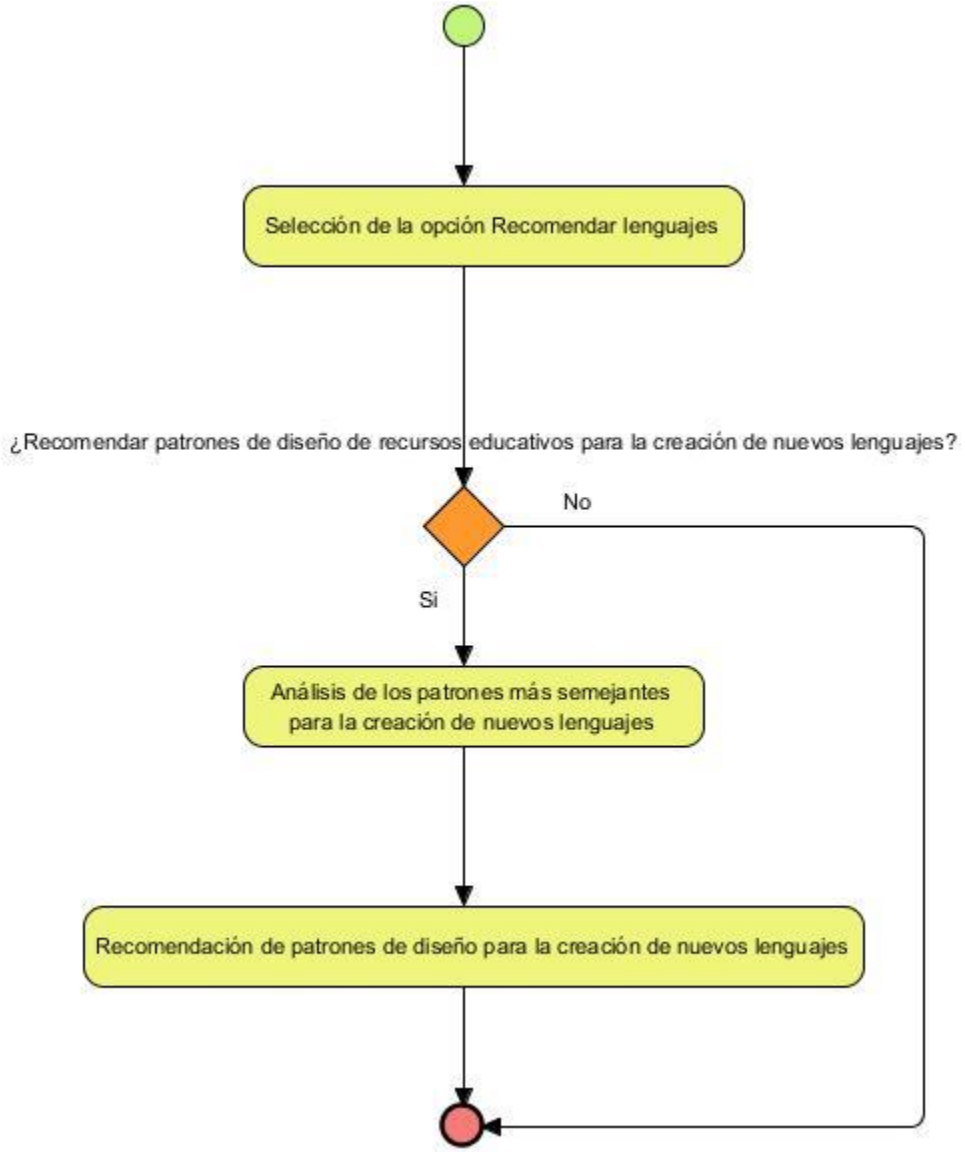


Ilustración 2. Diagrama de proceso del sistema para la opción recomendación de nuevos lenguajes

A continuación, se muestra un seudocódigo del principal proceso que realiza el sistema:

Función Recomendar nuevos lenguajes (): Colección

Inicio

Obtener patrones independientes

Obtener lenguajes

Obtener parones pertenecientes a catálogos

Inicializar colección vacía

Adicionar patrones independientes a la colección.

Adicionar patrones pertenecientes a catálogos a la colección

Adicionar lenguajes a la colección

Colección agrupamiento (tiempo en que fueron consultados)

Si tiempo en que fueron consultados < 15 dias

Adicionar elemento a lista auxiliar

Si elemento de la lista auxiliar (problema que resuelve) <> los elementos siguientes de la lista auxiliar (problema que resuelve)

Y <> los elementos siguientes de la lista auxiliar (solución)

Adicionar elemento a Colección recomendados

Ordenar elementos de Colección recomendados (Cantidad de descargas, cantidad de visualizaciones, evaluación)

Fin si

Fin si

2.2 Requisitos funcionales y no funcionales

El proceso de desarrollo de software comprende desde sus etapas más tempranas la definición de tareas orientadas a captar las necesidades o características del sistema que se vaya a crear o modificar[14, 55].

2.2.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de las funcionalidades que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones específicas. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos[56].

A continuación, se muestran los requisitos funcionales de la aplicación anterior, los cuales deberán ser modificados:

Nº	Nombre	Descripción	Prioridad	Complejidad
RF1	Establecer la conexión con los repositorios de patrones de diseño de recursos educativos.	Al iniciar el sistema, el mismo debe realizar la conexión con varios repositorios de patrones de diseño y de ser posible actualizar la base de datos local	Alta	Baja
RF2	Introducir el problema de recursos educativos.	Brindar la opción al usuario, para que mediante un campo de texto pueda introducir el problema de recurso educativo al cual debe dar solución.	Alta	Media
RF3	Realizar el análisis comparativo de los textos de los problemas introducido por el usuario y los problemas de los patrones pertenecientes a catálogos, lenguajes y patrones independientes almacenados en la base de datos local.	Una vez el usuario haya introducido el problema a resolver se le debe realizar el análisis comparativo del texto introducido con el texto de los problemas de los patrones de diseño independientes, lenguajes y patrones de diseño pertenecientes a catálogos almacenados en la base de datos local.	Alta	Media
RF4	Calcular el índice de interacción de los usuarios con los lenguajes, patrones pertenecientes a catálogos y patrones independientes,	Una vez que se identifiquen los patrones, lenguajes y patrones pertenecientes a catálogos que coincidan con el problema introducido por el usuario, se procede a	Alta	Media

	utilizando la calidad percibida.	calcular el índice de interacción de los usuarios. Utilizando para obtener este índice la calidad percibida por los usuarios.		
RF5	Listar los lenguajes y patrones de diseño obtenidos en el análisis comparativo.	Una vez el usuario haya obtenido el índice de interacción de los usuarios para cada uno de los resultados, se realiza un ordenamiento de mayor a menor teniendo en cuanta el valor del índice de interacción. Luego se debe mostrar a los usuarios el resultado obtenido en un listado.	Alta	Media
RF6	Mostrar la procedencia de los lenguajes, patrones pertenecientes a catálogos y patrones independientes recomendados.	Permitir mostrar al usuario la procedencia del lenguaje, patrón perteneciente a catálogos o patrón independiente listado que responda el problema introducido.	Alta	Media

A continuación, se muestran los requisitos funcionales necesarios para la implementación de las nuevas funcionalidades:

Nº	**Nombre**	**Descripción**	**Prioridad**	**Complejidad**
RF7	Agrupar patrones de diseño de recursos educativos que pudieran conformar lenguajes.	Una vez seleccionada la opción Recomendación de nuevos lenguajes, se procederá automáticamente a realizar	Alta	Alta

		el agrupamiento de patrones que pudieran conformar un nuevo lenguaje. Este agrupamiento se realizará realizado a partir de un algoritmo de agrupamiento, utilizando como entrada la fecha en que fueron visitados por los usuarios.		
RF8	Calcular el índice de interacción de los usuarios con los patrones agrupados, utilizando la calidad percibida.	Una vez que se agrupen los patrones de diseño de recursos educativos a partir de la utilización del algoritmo K-means, se procede a obtener el índice de interacción de los usuarios utilizando la calidad percibida. Luego el resultado obtenido se utilizará para ordenar los patrones, comenzando por los que posean un mayor índice.	Alta	Alta
RF9	Listar patrones de diseño de recursos educativos que pudieran conformar lenguajes.	Una vez que se ordenen los patrones resultantes del agrupamiento realizado, el sistema debe mostrar un listado con los posibles patrones que pudieran conformar un lenguaje.	Alta	Alta
RF10	Exportar en formato XML los patrones que pudieran conformar nuevos lenguajes.	Una vez listados los posibles patrones que pudieran conformar lenguajes, se le debe	Alta	Alta

		brindar al usuario la opción para poder exportar el resultado obtenido en un archivo XML.		

2.2.2 Requisitos no funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema[56].

Requisitos de Fiabilidad:

- La aplicación debe mostrar resultados exactos en sus operaciones, para credibilidad y seguridad del usuario.

Requisitos de hardware:

- El sistema para establecer la conexión con los repositorios de patrones de diseño se basa en la arquitectura SOA por lo que necesita un medio de comunicación. La Red LAN servirá para comunicar el sistema (aplicación de escritorio) con los repositorios.

Requisitos mínimos de hardware:

- Velocidad del procesador: 1Ghz
- Memoria RAM: 512 MB
- Disco duro: 1 GB disponible

2.3 Historias de usuarios

Como resultado de aplicar la metodología híbrida AUP en su versión UCI, específicamente en el escenario cuatro. La cual especifica que la forma de modelar los requisitos del sistema es a través de las HU. Dicha metodología propone que las HU deben ser redactadas por el cliente, el cual debe darles un nivel de prioridad. Por lo antes planteado para la solución propuesta se describen las siguientes HU:

Tabla 3. HU. Establecer la conexión con los repositorios de patrones de diseño de recursos educativos.

Número: 1	**Requisito:** Establecer la conexión con los repositorios de patrones de diseño de recursos educativos.
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 5 días
Riesgo en desarrollo: Inexistencia de repositorios *online* para consumir los servicios deseados.	**Tiempo real:** 4 días
Descripción: Cuando el usuario inicie la aplicación, el sistema debe establecer la conexión con los repositorios de los patrones de diseño, de ser posible actualizando la base de datos local.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 4. HU. Introducir el problema de diseño de recursos educativos.

Número: 2	**Requisito:** Introducir el problema de diseño de recursos educativos
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 3 días
Riesgo en desarrollo: El problema introducido por el usuario no es almacenado correctamente.	**Tiempo real:** 2 días
Descripción: El sistema debe permitir introducir el problema de recursos educativos. Cuando el usuario introduzca correctamente el problema de recursos educativos y seleccione la opción “Recomendar”, se le muestra el listado de lenguajes y patrones que responden a ese problema.	
Observaciones: Si el problema introducido es incorrecto se le mostrará un mensaje al usuario para notificarle que el problema es incorrecto, y se le dará la posibilidad al usuario de introducir nuevamente el problema.	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 5. HU. Realizar el análisis comparativo de los textos de los problemas introducidos por el usuario y los problemas de los patrones pertenecientes a catálogos, lenguajes y patrones independientes.

Número: 3	**Requisito:** Realizar el análisis comparativo de los textos de los problemas introducidos por el usuario y los problemas de los patrones pertenecientes a catálogos, lenguajes y patrones independientes, almacenados en la base de datos local.
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 7 días
Riesgo en desarrollo: Poca disponibilidad de patrones de recursos educativos almacenados.	**Tiempo real:** 5 días
Descripción: Cuando el usuario selecciona la opción “Recomendar patrones”, automáticamente serán analizados los patrones de diseño pertenecientes a catálogos, lenguajes y patrones	

independientes cuyos problemas coincidan con el problema introducido por el usuario. Luego los patrones perteneciente a catálogos, lenguajes y patrones independientes seleccionados, serán mostrados al usuario como una posible solución.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 6. HU. Calcular el índice de interacción de los usuarios con los lenguajes, patrones pertenecientes a catálogos y patrones independientes, utilizando la calidad percibida.

Número: 4	**Requisito:** Calcular el índice de interacción de los usuarios con los lenguajes, patrones pertenecientes a catálogos y patrones independientes, utilizando la calidad percibida
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 2 días
Riesgo en desarrollo: Poca disponibilidad de información almacenada en la base de datos.	**Tiempo real:** 2 días
Descripción: Cuando se hayan listado los patrones, lenguajes y patrones pertenecientes a catálogos que coincidan con el problema introducido por el usuario, el sistema automáticamente deberá proceder a calcular el índice de interacción de los usuarios para cada uno de los elementos seleccionados. Este índice de interacción será calculado a partir de la calidad percibida y se utilizará para ordenar dichos resultados, comenzando por los que posean mayor índice.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 7. HU. Listar los lenguajes y patrones de diseño obtenidos en el análisis comparativo.

Número: 5	**Requisito:** Listar los lenguajes y patrones de diseño obtenidos en el análisis comparativo.
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 3 días
Riesgo en desarrollo: N/A	**Tiempo real:** 2 días

Descripción: Luego de haber realizado la comparación entre textos y ordenamiento de los resultados, el sistema debe listar los lenguajes y patrones de diseño que brinden solución al problema introducido por el usuario.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 8. HU. Mostrar la procedencia de los lenguajes, patrones pertenecientes a catálogos y patrones independientes recomendados.

Número: 6	**Requisito:** Mostrar la procedencia de los lenguajes, patrones pertenecientes a catálogos y patrones independientes recomendados
Programador: David Grau Rodríguez	**Iteración asignada:** 1era
Prioridad: Alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: N/A	**Tiempo real:** 8 días
Descripción: Una vez que se le muestra al usuario el listado de los lenguajes y patrones de diseño que dan respuesta al problema introducido, se le brinda la opción al usuario de conocer la procedencia de cada uno de los resultados mostrados. Para conocer la procedencia debe seleccionar la opción “Mostrar procedencia”.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 9. HU. Agrupar patrones de recursos educativos que pudieran conformar nuevos lenguajes.

Número: 7	**Requisito:** Agrupar patrones de diseño de recursos educativos que pudieran conformar lenguajes
Programador: David Grau Rodríguez	**Iteración asignada:** 2da
Prioridad: Alta	**Tiempo estimado:** 6 días
Riesgo en desarrollo: Poca disponibilidad de información almacenada en la base de datos.	**Tiempo real:** 6 días
Descripción:	

Cuando el usuario seleccione la opción “Agrupar patrones”, el sistema automáticamente realiza un agrupamiento de los patrones almacenados en la base de datos. Dicho agrupamiento se realiza utilizando el algoritmo K-means de la librería Weka.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

Tabla 10. HU. Calcular el índice de interacción de los usuarios con los patrones agrupados, utilizando la calidad percibida.

Número: 8	**Requisito:** Calcular el índice de interacción de los usuarios con los patrones agrupados, utilizando la calidad percibida
Programador: David Grau Rodríguez	**Iteración asignada:** 2da
Prioridad: Alta	**Tiempo estimado:** 2 días
Riesgo en desarrollo: Poca disponibilidad de información almacenada en la base de datos.	**Tiempo real:** 2 días
Descripción: Luego que el algoritmo realiza el agrupamiento, los resultados serán ordenados teniendo en cuenta la calidad percibida por los usuarios al interactuar con cada uno de los patrones seleccionados. El ordenamiento será realizado comenzando por los patrones que mayor calidad percibida posean.	
Observaciones:	
Prototipo elemental de interfaz gráfica de usuario:	

Tabla 11. HU. Listar los patrones de diseño de recursos educativos que pudieran conformar un lenguaje.

Número: 9	**Requisito:** Listar los patrones de diseño de recursos educativos que pudieran conformar un lenguaje
Programador: David Grau Rodríguez	**Iteración asignada:** 2da
Prioridad: Alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: N/A	**Tiempo real:** 10 días
Descripción: Una vez realizado el agrupamiento y ordenados los resultados obtenidos en el mismo, el	

sistema debe mostrar al usuario un listado con los posibles patrones que pudieron conformar un lenguaje.

Observaciones:

Prototipo elemental de interfaz gráfica de usuario:

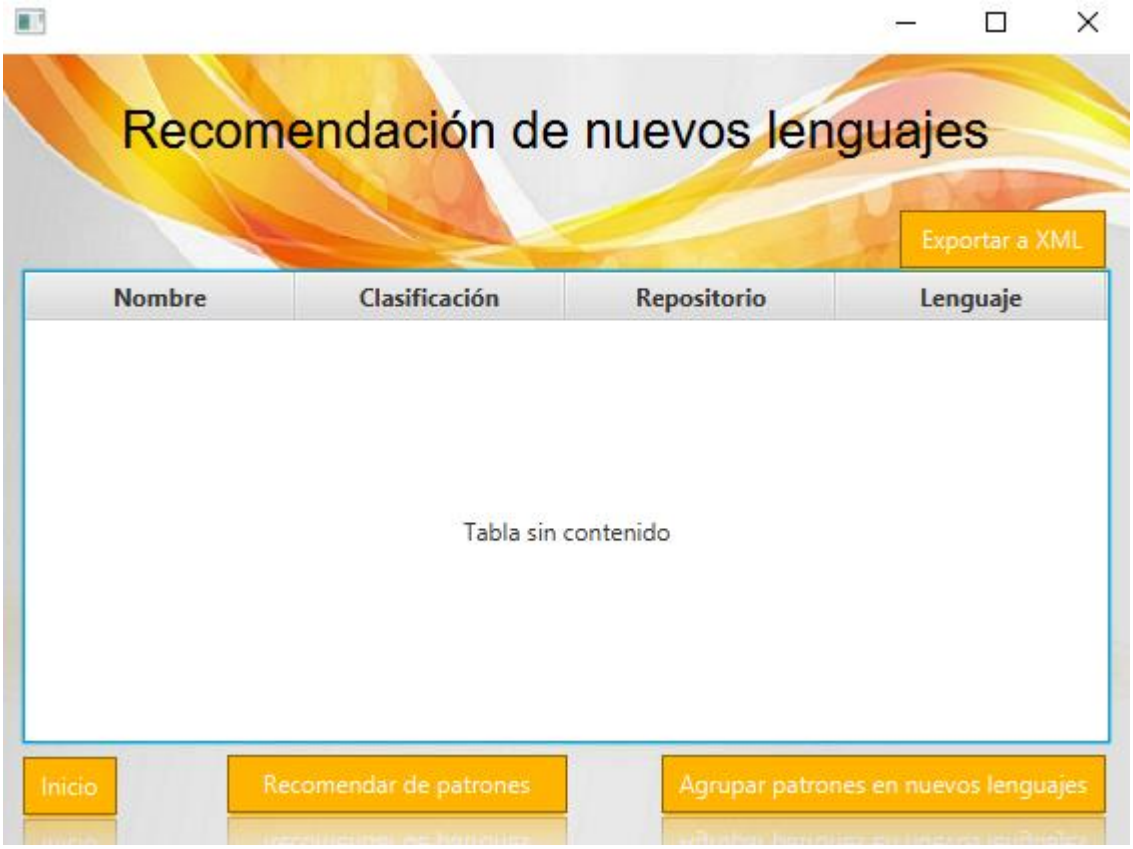


Tabla 12. HU. Exportar en formato XML los patrones que pudieran conformar nuevos posibles lenguajes.

Número: 10	**Requisito:** Exportar en formato XML los patrones que pudieran conformar posibles nuevos lenguajes
Programador: David Grau Rodríguez	**Iteración asignada:** 2da
Prioridad: Alta	**Tiempo estimado:** 10 días
Riesgo en desarrollo: N/A	**Tiempo real:** 10 días

Descripción:

Una vez listados los patrones de diseño de recursos educativos que pudieran conformar un posible lenguaje, el usuario tiene la opción de exportar la solución obtenida en un archivo con

formato XML, seleccionando el botón “Exportar”.
Observaciones:
Prototipo elemental de interfaz gráfica de usuario:

2.4 Diagrama de clases del análisis

Según Pressman en su libro “Un enfoque práctico” la clase de análisis constituye un artefacto fundamental del modelo de análisis[14]. Los diagramas de clases del análisis (DCA) se utilizan para mostrar las clases y sus relaciones[57].

A continuación, se muestra el diagrama de clases del análisis correspondiente a los RF del 2 al 15 que representan las operaciones correspondientes a la recomendación de lenguajes, patrones pertenecientes a catálogos y patrones de diseño de recursos educativos:

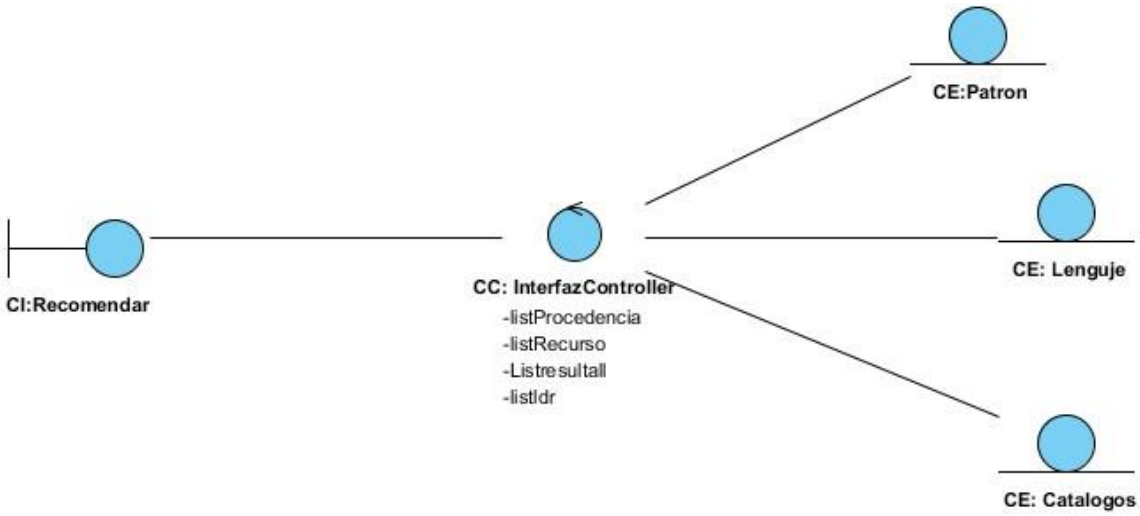


Ilustración 3. Diagrama de clases del análisis para los RF6, RF7 y RF8

A continuación, se muestra el diagrama de clases del análisis de correspondiente a los RF 15,16,17,18 que representan las operaciones correspondientes a la recomendación de posibles nuevos lenguajes:

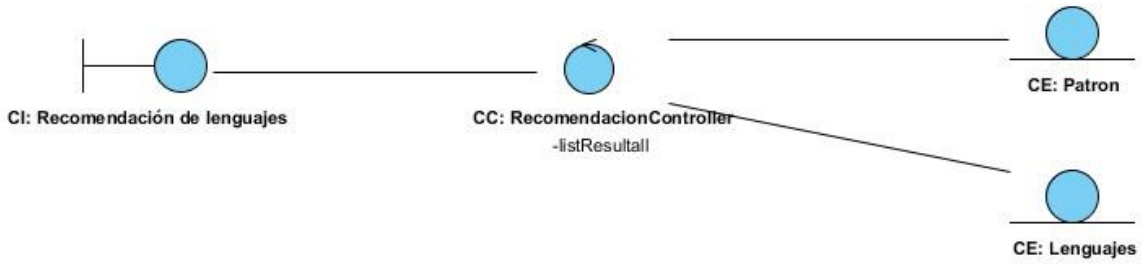


Ilustración 4. Diagrama de clases del análisis para los RF9, RF10 y RF11

2.5 Diagrama de clases del diseño

El diagrama de clases del diseño es una descripción gráfica las especificaciones de las clases de software y de las interfaces en una aplicación[58].

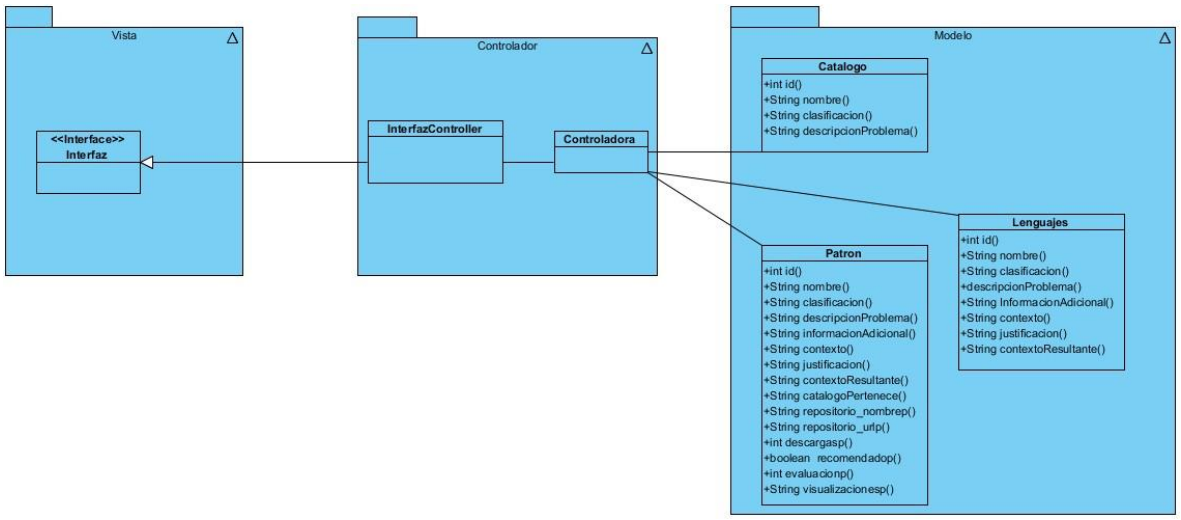


Ilustración 5. Diagrama de clases del diseño para los RF6, RF7 y RF8

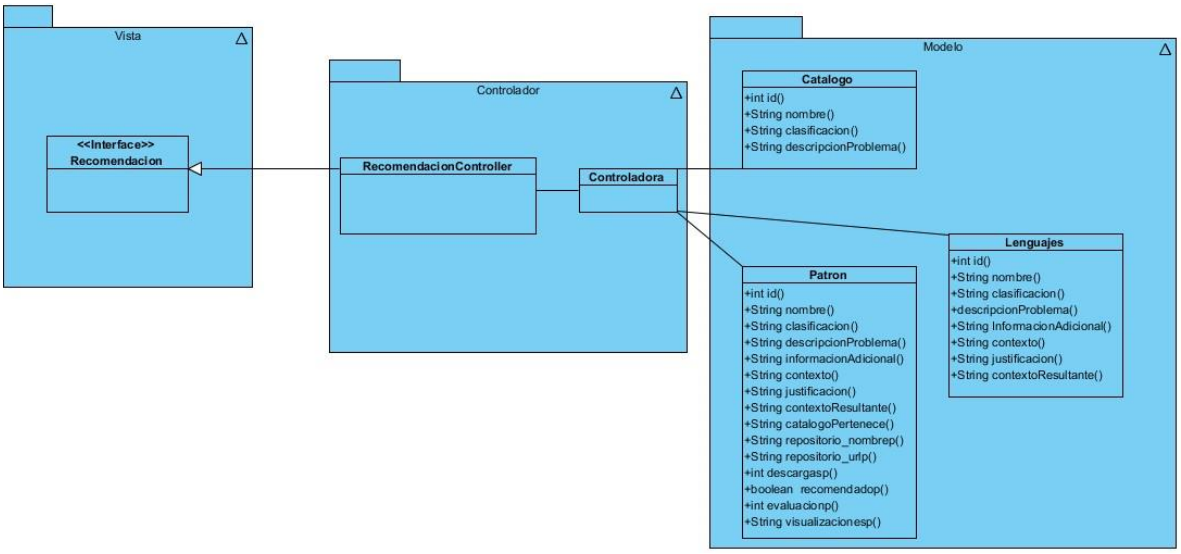


Ilustración 6. Diagrama de clases del diseño para los RF9, RF10 y RF11

2.6 Patrones de diseño a utilizar

Los patrones de diseño son aquellos que expresan esquemas para definir estructuras de diseño con las que construir sistemas de software.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular.

Un patrón de diseño está compuesto por tres aspectos fundamentales[56, 59]:

- **Contexto:** Las situaciones recurrentes.
- **Problema:** Metas y restricciones en el contexto.
- **Solución:** Diseño para conseguir las metas dentro de las restricciones.

2.6.1 Patrones GRASP

Los patrones GRASP General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades en sus siglas en inglés) describen los principios fundamentales para asignar responsabilidades a los objetos[60].

Principales patrones GRASP:

- **Experto:** es el patrón más utilizado en la asignación de responsabilidades, es un principio básico que suele utilizarse en diseño orientado a objetos. Con la utilización de este patrón se conserva el encapsulamiento, los objetos se valen de

su propia información para hacer lo que se les pide. El comportamiento (de los objetos) se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas que son más fáciles de comprender y mantener. Se evidencia el patrón en la solución propuesta en las clases PrincipalController, InterfazController y RecomendacionController.

```
public class PrincipalController implements Initializable {

    /**
     * Initializes the controller class.
     */
     @FXML
    private Button button1;
     @FXML
    private Button button3;
    private  Stage stage;


    public void setStage(Stage stage) {
        this.stage = stage;
    }
```

Ilustración 7. Implementación del patrón Experto en el sistema.

- **Creador:** El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Se evidencia el patrón en la solución propuesta en las clases InterfazController y RecomendacionController.

```
rs.next();
Patron patr = new Patron(rs.getInt("id"),
        rs.getString("nombre"),
        rs.getString("clasificacion"),
        rs.getString("descripcionproblema"),
        rs.getString("informacionadicional"),
        rs.getString("contexto"),
        rs.getString("justificacion"),
        rs.getString("contextoresultante"),
        rs.getString("catalogopertenece"),
        rs.getInt("lenguaje"),
        rs.getString("repositorio_nombrep"),
        rs.getString("repositorio_urlp"),
        rs.getInt("descargasp"),
        rs.getBoolean("recomendadop"),
        rs.getInt("evaluacionp"),
        rs.getInt("visualizacionesp"));

//  System.out.println(patr);
pac.add(patr);
```

Ilustración 8. Implementación del patrón Creador en el sistema.

- **Alta Cohesión:** Con el uso de este patrón mejoran la claridad y la facilidad con que se entiende el diseño. Se simplifican el mantenimiento y las mejoras en funcionalidad. A menudo se genera un bajo acoplamiento. Soporta una mayor capacidad de reutilización, porque una clase muy cohesiva puede destinarse a un propósito muy específico. Este patrón se puede observar en la clase Controller, ya que estas funcionalidades son moderadas, están en su área funcional y colabora con otras clases como Lenguajes.java, Patrones.java y Catalogos.java.

- **Bajo Acoplamiento:** Es un patrón evaluativo que el diseñador aplica al juzgar sus decisiones de diseño. Estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecienten la oportunidad de una mayor productividad. Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar. Este patrón se evidencia en la clase InterfazController, la cual contiene una instancia de

la clase Controller, por lo que toma todas las responsabilidades de la misma evitando una sobrecarga de relaciones con otras clases.

- **Controlador:** Garantiza que la empresa o los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la interfaz. Al delegar a un controlador la responsabilidad de la operación de un sistema entre las clases del dominio favorece la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras. Se evidencia el patrón en la solución propuesta en las clases PrincipalController, InterfazController y RecomendacionController.

```
Stage stage= new Stage();
FXMLLoader loader = new FXMLLoader(getClass().getResource("Recomendacion.fxml"));
Parent root = loader.load();
RecomendacionController controller = loader.getController();
controller.setStage(stage);
stage.setScene(new Scene(root));
stage.show();
this.stage.hide();
```

Ilustración 9. Implementación del patrón Controlador en el sistema.

2.6.2 Patrones GoF

Los patrones GoF fueron introducido popularmente a la industria del software con el libro Design Patterns: Elements of reusable object oriented programming escrito por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides en 1995. Estos introducen estructuras de clases e interfaces que solucionan un problema general, pensado para que las aplicaciones robustas sean escalables y flexibles, además rápidas de desarrollar y diseñar. Esta serie de patrones permiten ampliar el lenguaje y aprender nuevos estilos de diseño[61, 62].

Los autores dividen los patrones GoF en tres principales grupos[62, 63]:

- **Patrones de diseño estructurales:** se enfocan en el proceso de instanciación y esto proporciona la ventaja de depender más en la composición de objetos que la herencia de clases.
- **Patrones de diseño de comportamiento:** Se enfocan en facilitar el diseño, al identificar formas sencillas de relacionar entidades. Estos patrones describen

formas de componer objetos en caso de agregar nuevas funcionalidades a dichos objetos.

- **Patrones de diseño creacionales:** Los patrones de diseño de comportamiento se enfocan en la comunicación de los objetos de las clases.

De los patrones GoF a utilizar en la presente investigación es el patrón de diseño de comportamiento Facade este patrón provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

Patrón facade (Fachada)

Problema:

Se requiere una interfaz común para un conjunto de implementaciones o interfaces dispares.

Solución:

Defina un único punto de conexión de un subsistema, este objeto Fachada presenta una única interfaz unificada y es responsable de colaborar con los clientes. (Controlador de fachada).

Consecuencias:

- Oculta a los clientes los componentes del subsistema.
- Promueve acoplamiento débil entre el subsistema y los clientes.
- No impide que las aplicaciones usen las clases del subsistema.

La utilización de este patrón se evidencia en la vista Recomendación.fxml, la cual propone al usuario una interfaz única, pero que interactúa con diversas clases del sistema.

2.7 Conclusiones del capítulo

En el presente capítulo se describieron los requisitos funcionales y no funcionales identificados para el desarrollo del sistema. Se logró realizar un correcto diseño del sistema, como resultado de la utilización de los diagramas de clases del análisis y los diagramas de clases del diseño. A través de la utilización de los patrones GRASP y GoF, se logró realizar una adecuada implementación del sistema posibilitando una organización adecuada del código de la aplicación.

Capítulo 3. Implementación y validación del sistema inteligente

En el presente capítulo se detallan los métodos implementados para el cumplimiento del problema trazado en la investigación. Además, se evidencian los tipos de pruebas empleadas para validar el sistema y garantizar la calidad del mismo. Estos contaron con una estrategia de prueba, teniendo en cuenta los niveles de pruebas existentes. Dichos niveles de pruebas fueron aplicados al sistema a partir, del uso de métodos de prueba tales como: el método de caja blanca y el método de caja negra.

3.1 Implementación

En ciencias de la computación, una implementación es la realización de una especificación técnica o algoritmos como un programa, componente de software u otro sistema de cómputo. Muchas implementaciones son dadas según a una especificación o un estándar.

La etapa de implementación consistirá en la codificación del sistema inteligente partiendo de los artefactos generados en la extracción de requisitos y elaboración del diseño.

3.1.1 Estándares de codificación

Los estándares de codificación son reglas internacionales que regulan la forma de implementación del código de programación, logrando una homogeneidad. Los estándares de codificación facilitan el proceso de creación del sistema, así como los procesos de mantenimiento del mismo, independientemente del autor.

En la implementación del sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes potenciales se utilizaron varios estándares para favorecer la legibilidad y organización del código[64].

Convenciones de codificación para la nomenclatura[65]:

- Las variables de clase o método y las instancias deben declararse en formato camelCase, empezar por letra minúscula y, si se trata de palabras concatenadas, la primera letra de las siguientes palabras en mayúscula.

```
public class Patron {
    private int id;
    private String nombre;
    private String clasificacion;
    private String descripcionProblema;
    private String informacionAdicional;
    private String contexto;
    private String justificacion;
    private String contextoResultante;
    private String catalogoPertenece;
    private String repositorio_nombrep;
    private String repositorio_urlp;
    private int descargasp;
    private boolean  recomendadop;
    private int evaluacionp;
    private String visualizacionesp;
```

Ilustración 10. Utilización de las convenciones de codificación de la nomenclatura para variables clases o métodos.

- Las variables constantes se deben declarar totalmente en mayúsculas, separando las palabras con el guión bajo (“_”) si se trata de una palabra compuesta.
- Las clases deben tener la primera letra mayúscula al igual que la primera letra de las siguientes palabras si se trata de un nombre compuesto. Se recomienda que los identificadores de las clases sean sustantivos.

```
public class Lenguaje {

public class Catalogo {
```

Ilustración 11.Utilización de las convenciones de codificación de nomenclaturas para clases.

3.1.2 Funcionalidades implementadas

En el presente epígrafe se hace alusión a los métodos más relevantes implementados, detallando el funcionamiento de los mismos.

Para lograr un correcto funcionamiento de los algoritmos de la librería Weka es necesario que los datos de entrada estén en un archivo con formato (“arff” o “csv”). Por tal motivo, a continuación, se presenta el tratamiento realizado a la base de datos local para guardar la información en un archivo “csv”.

```
Statement csvf = null;
String csv = "COPY visita TO 'C:\\Windows\\Temp\\visita.csv'' delimiters ';' WITH CSV HEADER;";

Connection dbConection = null;
Class.forName("org.postgresql.Driver");
String direccion = "localhost";
String db_address = "jdbc:postgresql://" + direccion + ":5432/pattern";
dbConection = DriverManager.getConnection(db_address, "postgres", "1");

csvf = dbConection.createStatement();
csvf.execute(csv);
```

Ilustración 12. Código para importar el contenido de la base de datos local a un archivo "csv".

A continuación, se presenta la implementación del algoritmo SimpleKmeans de la librería Weka. Este algoritmo recibe como entrada la información de almacenada en el archivo “csv”, la cantidad de clusters en los que debe agrupar la información y la cantidad de iteraciones que se deben realizar del algoritmo.

```
int[] resCluster = wk.run(args);
Class.forName("org.postgresql.Driver");
dbConection = DriverManager.getConnection(db_address, "postgres", "1");
Statement statement2 = dbConection.createStatement();

LinkedList<Patron> pac = new LinkedList<Patron>();
String select = "select p.* from visita v inner join patron p   on(v.idpatron=p.id) ";
ResultSet rs = statement2.executeQuery(select);

do {

    rs.next();
    Patron patr = new Patron(rs.getInt("id"),
            rs.getString("nombre"),
            rs.getString("clasificacion"),
            rs.getString("descripcionproblema"),
            rs.getString("informacionadicional"),
            rs.getString("contexto"),
            rs.getString("justificacion"),
            rs.getString("contextoresultante"),
            rs.getString("catalogopertenece"),
            rs.getInt("lenguaje"),
            rs.getString("repositorio_nombrep"),
            rs.getString("repositorio_urlp"),
            rs.getInt("descargasp"),
            rs.getBoolean("recomendadop"),
            rs.getInt("evaluacionp"),
            rs.getString("visualizacionesp"));

    //  System.out.println(patr);
    pac.add(patr);
```

Ilustración 13. Implementación del algoritmo SimpleKmeans de la librería Weka.

Luego del agrupamiento realizado con el algoritmo K-means se hace necesario ordenar los resultados obtenidos, con el objetivo de brindarle al usuario un listado ordenado,

comenzando por los patrones que cuentes con un mayor indicador de calidad percibida. Por tal motivo, a continuación, se presenta la implementación realizada del método CalidadPercivida ().

```
public double CalidadPercivida(int d, int e,boolean r, int v){

        double ev=0;

    int r1=0;
    if(r)
      r1=1;

    int cont=0;
    double sum=0;
    while (cont<v){
        sum+=(double) ((0.5*(e/5)+0.3*r1+0.2*d)/v);

    cont++;
    }
         return sum;
     }
}
```

Ilustración 14. Implementación del algoritmo de calidad percibida

Luego de haber realizado el agrupamiento el ordenamiento correspondiente, a de la calidad percibida por el usuario. Teniendo en cuenta que el proyecto GITAE al cual tributa este sistema, establece que los sistemas desarrollados en el mismo deben establecer una conexión de sus resultados a partir de archivos XML. Se le brinda la posibilidad al usuario de exportar el agrupamiento obtenido en un archivo con dicho formato. A continuación, se presenta una imagen con la implementación de esta funcionalidad.

```
public void crearXML(ObservableList<Nuevoslenguajes> resullAll) {
    try {
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
        //Elemento raíz
        Document doc = docBuilder.newDocument();
        Element rootElement = doc.createElement("root");
        doc.appendChild(rootElement);

        for (int i = 0; i < resullAll.size(); i++) {

            //Primer elemento
            Element elemento1 = doc.createElement("id");
            rootElement.appendChild(elemento1);
            //Se agrega un atributo al nodo elemento y su valor
            Attr attr = doc.createAttribute("id");
            attr.setValue(""+resullAll.get(i).getId());
            elemento1.setAttributeNode(attr);

            Element elemento2 = doc.createElement("nombre");
            //elemento2.setTextContent(""+resullAll.get(i).getNombre());
            rootElement.appendChild(elemento2);
            Attr attr2= doc.createAttribute("nombre");
            attr2.setValue(resullAll.get(i).getNombre());
            elemento2.setAttributeNode(attr2);

            Element elemento3 = doc.createElement("clasificacion");
            elemento3.setTextContent(""+resullAll.get(i).getClasificacion());
            rootElement.appendChild(elemento3);

             Element elemento4 = doc.createElement("repositorio");
             elementc4.setTextContent(""+resullAll.get(i).getRepositorio_nombrep());
             rootElement.appendChild(elemento4);

             Element elemento5 = doc.createElement("visita");
             elementc5.setTextContent(""+resullAll.get(i).getVisita());
             rootElement.appendChild(elemento5);
        }
        //Se escribe el contenido del XML en un archivo
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        DOMSource scurce = new DOMSource(doc);
        //StreamResult result = new StreamResult(new File("/ruta/prueba.xml"));
        FileChooser chooser = new FileChooser();
        chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Archivo .xml", "*.xml"));
        File file = chooser.showSaveDialog(stage);
        if (file != null) {
            StreamResult result = new StreamResult(file);
            transformer.transform(source, result);
        }
    } catch (ParserConfigurationException pce) {
        pce.printStackTrace();
    } catch (TransformerException tfe) {
        tfe.printStackTrace();
    }
```

Ilustración 15. Implementación del algoritmo para importar en formato XML

Una vez finalizada la fase de implementación, es necesario realizar las pruebas para comprobar el correcto funcionamiento de los algoritmos.

3.2 Pruebas

Las pruebas son una actividad, en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, los resultados son observados y registrados para una posterior evaluación. Son un elemento de suma importancia para la calidad del software, por lo que se llevan a cabo durante todo el ciclo de vida del mismo. Constituyen una revisión final de las especificaciones del diseño y de la codificación[14].

Para realizar un correcto desarrollo de las pruebas es necesario tener en cuenta los tipos de pruebas existentes. Estos tipos de pruebas están divididos en dos grandes grupos, las pruebas funcionales y las pruebas no funcionales. La utilización de estos tipos de prueba, cuenta con una estrategia de pruebas definidas. Dichas estrategias cuentan con varios niveles de pruebas, necesarios para garantizar la calidad del sistema desarrollado. Los niveles de pruebas están conformados por las siguientes pruebas[14, 66]:

- Pruebas unitarias
- Pruebas del sistema
- Pruebas de regresión
- Pruebas de aceptación

Niveles de pruebas

Pruebas unitarias: consiste en la verificación de unidades del software de forma aislada, es decir, probar el correcto funcionamiento de una unidad de código. Generalmente, se realizan pruebas de caja blanca para comprobar que el código se corresponde con las especificaciones del componente[66, 67].

Pruebas de sistema: se realizan una vez que se han probado los componentes y la integración de los mismos, con el objetivo de estudiar los requisitos funcionales y no funcionales del sistema y las características de calidad. Para ello, se aplican técnicas que se corresponden con el método de caja negra[66, 67].

Pruebas de aceptación: son responsabilidad del cliente y pueden ser la única parte de las pruebas en donde estén involucrados. Se llevan a cabo antes de que el programa se ponga en funcionamiento real y tienen que satisfacer las expectativas del cliente[66, 67].

Las pruebas de aceptación a menudo se realizan en dos etapas:

Alfa: se lleva a cabo por el cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado[66, 68].

Beta: se realizan con posterioridad a las pruebas alfa, y se desarrollan en el entorno del cliente. En este caso, el cliente se queda a solas con el producto y trata de encontrarle fallos de los que informa al desarrollador[66, 68].

Para el desarrollo de estos niveles de pruebas, son utilizados varios métodos de pruebas. Los métodos de prueba de software incluyen la identificación, medición y evaluación de una o más cualidades, características o propiedades del software en cuestión. En las pruebas realizas al presente sistema se utilizaron dos métodos de pruebas: los métodos de caja blanca y los de caja negra. A continuación, se describen ambos métodos.

3.2.1 Método de caja blanca

El método de caja blanca, examinan la parte interna del programa, siempre se está observando el código. Se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa. Por ello, la implementación de estas pruebas depende de la disponibilidad del código fuente[69-71].

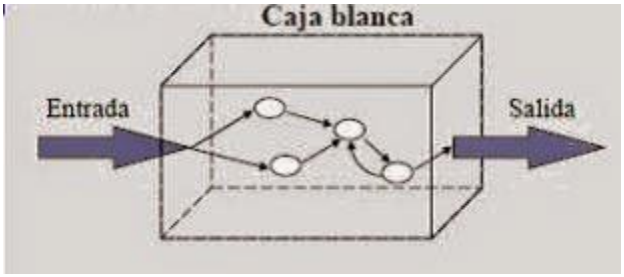


Ilustración 16. Método de caja blanca

Esta prueba se realiza durante toda la etapa de implementación del sistema, para poder probar y ejecutar el código generado.

Técnica de camino básico

Esta técnica permite obtener una medida de la complejidad lógica de un diseño procedimental y utiliza esa medida como guía para la definición de un conjunto básico de caminos de ejecución, de los cuales se obtienen los casos de prueba, que garantizan la ejecución de cada sentencia del programa al menos una vez, durante las pruebas[14].

Su ejecución se describe en cuatro pasos[14]:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado: se utiliza para representar el flujo de control lógico de un programa.
- Se calcula la complejidad ciclomática del grafo: el valor calculado define el número de rutas independientes en el conjunto básico de un programa, y proporciona un límite superior para el número de pruebas que deben aplicarse, lo cual asegura que todas las instrucciones se hayan ejecutado por lo menos una vez.
- Se determina un conjunto básico de caminos independientes: es cualquier ruta del programa que ingresa por lo menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

A continuación, se muestra la aplicación de la técnica de camino básico, al método crearXML () de la clase RecomendacionController, el cual tiene como objetivo, exportar en formato XML el resultado obtenido en la agrupación de patrones de diseño de recursos educativos en posibles lenguajes. Este método se eligió teniendo en cuenta la importancia que se le atribuye en el grupo de investigación GITAE, a la comunicación de las aplicaciones a través del formato XML.

Tabla 13. Definición de los nodos de flujo en el método crearXML ()

crearXML()	
0	public void crearXML(ObservableList<Nuevoslenguajes> resullAll) {

1	try {
	DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
	DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
	Document doc = docBuilder.newDocument();
	Element rootElement = doc.createElement("root");
	doc.appendChild(rootElement);

```
2  for (int i = 0; i < resullAll.size(); i++) {
3  Element elemento1 = doc.createElement("id");
   rootElement.appendChild(elemento1);
   Attr attr = doc.createAttribute("id");
   attr.setValue(""+resullAll.get(i).getId());
   elemento1.setAttributeNode(attr);
4  Element elemento2 = doc.createElement("nombre");
   rootElement.appendChild(elemento2);
   Attr attr2= doc.createAttribute("nombre");
   attr2.setValue(resullAll.get(i).getNombre());
   elemento2.setAttributeNode(attr2);
5  Element elemento3 = doc.createElement("clasificacion");
   elemento3.setTextContent(""+resullAll.get(i).getClasificacion());
   rootElement.appendChild(elemento3);
6
7  Element elemento4 = doc.createElement("repositorio");
   elemento4.setTextContent(""+resullAll.get(i).getRepositorio_nombrep());
   rootElement.appendChild(elemento4);
8  Element elemento5 = doc.createElement("visita");
   elemento5.setTextContent(""+resullAll.get(i).getVisita());
   rootElement.appendChild(elemento5);
   }
9  TransformerFactory transformerFactory = TransformerFactory.newInstance();
   Transformer transformer = transformerFactory.newTransformer();
   DOMSource source = new DOMSource(doc);
```

```
                FileChooser chooser = new FileChooser();
                chooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Archivo .xml",
"*.xml"));
                File file = chooser.showSaveDialog(stage);
10              if (file != null) {
                StreamResult result = new StreamResult(file);
                transformer.transform(source, result);
                }
11              } catch (ParserConfigurationException pce) {
                pce.printStackTrace();
12              } catch (TransformerException tfe) {
                tfe.printStackTrace();
                }
                }
```

A continuación, la figura muestra cómo queda conformado gráficamente el flujo de control lógico del método anterior.

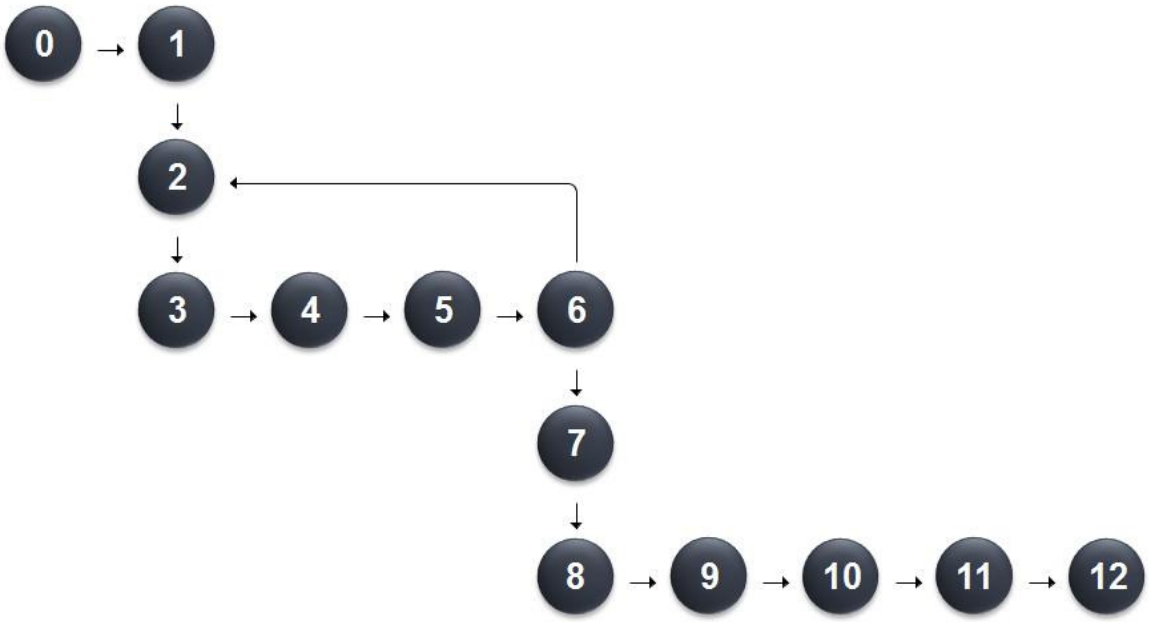


Ilustración 17. Flujo de control lógico correspondiente al método crearXML ()

Una vez construido el grafo, se procede a calcular la complejidad ciclomática usando la fórmula descrita a continuación.

Tabla 14. Complejidad ciclomática correspondiente al método crearXML ()

V(G) = A - N + 2
A = 13 (aristas) N = 12 (nodos)
V(G) = 13-13+2
V(G) = 2

Conociendo que **V(G)= 2**, se puede afirmar que se necesitan 2 caminos distintos para recorrer todos los nodos del grafo, donde cada camino representa una prueba a realizar para cada sentencia de código se ejecute al menos una vez. A continuación, los caminos determinados:

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
- 0, 1, 2, 3, 4, 5, 6, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

A cada uno de estos caminos le corresponde un caso de prueba, como se muestra en la siguiente tabla:

Tabla 15. Caso de prueba de los caminos independientes

No	Entrada	Resultado
1	El resultado del agrupamiento del algoritmo, el cual solo presenta un elemento por *clusters*.	Se crea exitosamente el archivo XML.
2	El resultado del agrupamiento realizado en el agrupamiento, el cual presenta varios elementos por cada *clusters*.	Se crea exitosamente el archivo XML, el cual contiene todos los elementos agrupados en los distintos *clusters*.

Resultados del método de caja blanca

Se aplicó la técnica de camino básico al código fuente del proceso, que demostraron el buen funcionamiento de los requisitos funcionales, alcanzándose las respuestas esperadas. Como resultados de esta técnica, se detectó una no conformidad en la primera iteración. En la segunda iteración se corrigió la no conformidad detectada en la iteración anterior y no fueron detectadas más no conformidades.

3.2.2 Método de caja negra

El método de caja negra, se enfoca en probar el sistema sin tomar en cuenta la estructura interna del mismo, su objetivo es validar que el comportamiento de las salidas del sistema sean las esperadas. Si las salidas no son las esperadas entonces la prueba ha detectado un problema con el software. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar [69-71].

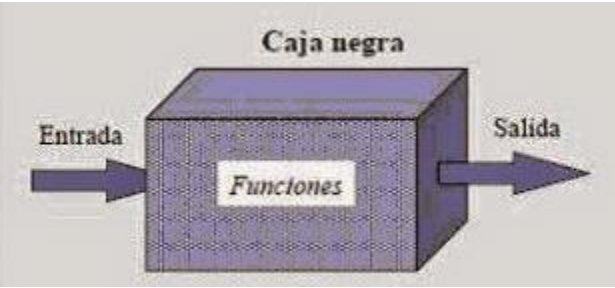


Ilustración 18. Método de caja negra

Para el empleo de este método de prueba, se utiliza una de las técnicas propuesta por ella la prueba de partición equivalente.

Técnica de partición equivalente

La partición de equivalencia es un método de prueba de Caja Negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Básicamente, este método intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia, de tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase. Esto quiere decir que, si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error, y viceversa; si un caso de prueba no ha detectado ningún error, es de esperar que ninguno de los casos de prueba correspondientes a la misma clase de equivalencia encuentre ningún error.

El diseño de casos de prueba según esta técnica consta de dos pasos[71]:

- Identificar las clases de equivalencia (valores para los cuales se espera que el programa tenga un comportamiento común).
- Identificar los casos de prueba.

Descripción de variables

En la siguiente tabla se muestra la variable de entrada con que cuenta el sistema inteligente.

Tabla 16. Descripción de variable

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Problema	Campo de texto	No	Campo de carácter obligatorio que representa el usuario, puede contener letras y

				números.

Casos de prueba

En el presente epígrafe se detallan algunos de los casos de prueba correspondientes a los requisitos funcionales identificados en la investigación.

Requisito: "Introducir el problema de diseño de recursos educativos"

Descripción general: El requisito comienza cuando el usuario selecciona la opción "Recomendación de patrones", mostrándole una ventana para que el usuario introduzca el problema de recurso educativo y culmina cuando el usuario presiona el botón "Recomendar".

Tabla 17. Caso de prueba FR. Introducir el problema de diseño de recursos educativos

Escenario	Descripción	Problema	Respuesta del sistema	Flujo Central
EC 1.1 Selecciona la opción "Recomendación de patrones"	Muestra una ventana con la opción de introducir un campo de texto	N/A	Muestra los datos que se deben introducir para realizar la recomendación	Inicio/ Recomendación de patrones
EC 1.2 Selecciona la opción "Recomendar"	Introduce los datos y selecciona la opción "Recomendar"	V	Valida los datos introducidos	Inicio/ Recomendación de patrones / Recomendar
EC 1.3 Existen campos vacíos o datos incorrectos	Existen campos vacíos o datos incorrectos	I	Muestra un mensaje informando que debe introducir el	Inicio/ Recomendación de patrones

			problema	
EC 1.4 Selecciona la opción “atrás”	Se selecciona la opción “atrás”	N/A	Se muestra la página principal	Inicio/ Recomendación de patrones / Atrás
EC 1.5 Selecciona la opción “Recomendación de nuevos lenguajes”	Selecciona la opción “Recomendación de nuevos lenguajes”	N/A	Se muestra la vista Recomendación de nuevos lenguajes	Inicio/ Recomendación de patrones / Recomendación de nuevos lenguajes

3.3 Conclusiones del capítulo

En el desarrollo de este capítulo fueron descritas las nuevas funcionalidades implementadas al sistema. Para garantizar la calidad del sistema desarrollado se aplican las pruebas correspondientes. Estas pruebas se aplicaron mediante los métodos de caja blanca y caja negra. Para la aplicación del método de caja blanca se utilizó la técnica de camino básico. En el caso del método de caja negra se utilizó la técnica de partición equivalente. Como resultado significativo, en la primera iteración de las pruebas se obtuvo una no conformidad que fue corregida en la posterior iteración. Se validó la correcta implementación y el funcionamiento satisfactorio del Sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes.

Conclusiones generales

Con el desarrollo de la presente investigación, se dio cumplimiento a los objetivos propuestos inicialmente, por lo que se puede arribar a las siguientes conclusiones:

- El estudio bibliográfico de los conceptos características y antecedentes de los sistemas de recomendación y los algoritmos de agrupamiento, permitió seleccionar el algoritmo K-means para realizar el agrupamiento de patrones y como tipo de sistema de recomendación el basado en filtrado colaborativo.
- Se logró realizar un correcto diseño del sistema, como resultado de la utilización de los diagramas de clases del análisis y los diagramas de clases del diseño.
- Se implementó el sistema inteligente para el agrupamiento de patrones de diseño de recursos educativos en lenguajes potenciales, a partir de la estrategia de vecindad próxima y la calidad percibida por los usuarios.
- Se realizaron pruebas al sistema implementado, a partir de la utilización de las técnicas de caja blanca y caja negra, donde se constató su correcto funcionamiento.

Recomendaciones

Para la continuidad del trabajo desarrollado el autor recomienda:

- Desarrollar funcionalidades que permitan realizar el agrupamiento de patrones de diseño de recursos educativos en catálogos.
- Implementar un mecanismo para cosechar patrones en repositorios que no brinden servicios web.

Bibliografía

1. UNESCO. *Las TIC en la educación*. 2017 14/10/2017]; Available from: http://es.unesco.org/themes/tic-educacion.
2. Castillo, W.A. and A.B. Medina, *Aplicación de recursos didácticos para la mejora de la competencia actúa y piensa matemáticamente en situaciones de forma en el área de matemática en los estudiantes del segundo grado de educación secundaria de la institución educativa Daniel Becerra Ocampo*. 2017, Universidad nacional de San Agustín de Arequipa. p. 103.
3. Zapata, M., *Recursos educativos digitales: conceptos básicos*. 2012, Universidad de Antioquia.
4. Ortiz, Y., *Recursos Educativos Digitales que aportan al proceso de enseñanza y aprendizaje.* 2017. **3**.
5. Sánchez, N.C., *Sistema de recomendación de patrones de diseño de recursos educativos que incluye el tratamiento de agrupaciones y la calidad percibida por los usuarios*. 2018, Universidad de las Ciencias Informáticas. p. 85.
6. Alexander, C., *A pattern language*. Vol. 2. 1977.
7. HillsideGroup. *About patterns*. 2017 21/11/2017]; Available from: http://hillside.net/patterns/about-patterns.
8. Marciszack, M., et al., *Implementación de patrones en la validación de modelos conceptuales.* 2015.
9. Educación, M.d. *Patrones de diseño aplicados al desarrollo de Objetos Digitales Educativos*. 2017 27/09/2017]; Available from: http://ares.cnice.mec.es/informes/21/index.htm.
10. Gómez, Y.A. and Y.M. Martínez, *Sistema de recomendación de patrones de diseño para Recursos Educativos Abiertos*. 2015, Universidad de las Ciencias Informáticas. p. 73.
11. Ros, M.Z., *Patrones en elearning. Elementos y referencias para la formación.* Revista de Educación a distancia, 2011: p. 10.
12. Jiménez, J.M.R., *Patrones pedagógicos en educación virtual.* 2009 p. 16.
13. Montero, S., et al., *Patrones de diseño aplicados al desarrollo de objetivos digitales educativos*, M.d. educación and G.d. España, Editors. 2011.
14. Pressman, R.S., *Ingeniería del software. Un enfoque práctico*, ed. Edición. 2010.
15. Caro, M.F., J. Hernandez, and J.A. Jiménez, *Diseño de un sistema de recomendación en repositorios de objetos de aprendizaje basado en la percepción del usuario: Caso RODAS.* 2011.
16. Cañizares, R., *Repositorio de Recursos Educativos para las Instituciones de Educación Superior.* 2012.
17. Maya, L.S.A. and L.V. Useche, *Materiales compuestos inteligentes.* 2004.
18. Barroso, G.T., *El edificio como un sistema de gestión de información*, in *Facultad de informática*. 2003, Universidad Politécnica de Madrid.
19. Martínez, M.C., *Sistema de recomendación basado en técnicas de predicción de enlaces para los jueces en línea*, in *Facultad de Informática*. 2017, Universidad Complutense de Madrid. p. 128.
20. Vera, J., A.O. Mamani, and K. Villalba, *Modelo de sistema de recomendación de Objetos de Aprendizaje en dispositivos móviles, caso: Desarrollo del pensamiento computacional* 2015, Universidad Nacional de San Agustín
21. Crespo, M.A.M., *Sistema de recomendación híbrido para la predicción de calificaciones en Yelp.com*. 2016, Universidad politécnica de Madrid.
22. Adomavicius, G. and A. Tuzhilin, *Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions.* 2005.
23. Pepa, S.M., *Suite de algoritmos de recomendación en aplicaciones reales.*, in *Escuela politécnica superior*. 2014, Universidad Autónoma de Madrid.
24. Torres, E.J.C., *Algoritmos colaborativos para orientar académicamente al alumno en el bachillerato*, in *Departamento de informática*. 2007, Universidad de Jaén.

25. Rodrigo, J.A. *Sistema de recomendación con R*. 2018; Available from: https://rpubs.com/Joaquin_AR/370301.
26. Duran, D.F. and J.L.A. Herrera, *Algoritmo de filtrado híbrido-Mixto para recomendación de contenidos audiovisuales a comunidades virtuales*. 2013, Universidad del Cauca, Popayán, Colombia.
27. Gónzalez, Y.R., N.M. Sánchez, and M.M.G. Lorenzo, *Estructuración de la base de conocimiento en un sistesma basado en casos utilizando algorítmos conceptuales.* Revista Cubana de Ciencias Informáticas, 2018. **12**: p. 47-62.
28. Calatayud, R.E.L., *Un enfoque híbrido para la calificación de imágenes de resonancia magnética del cerebro*. 2016, Universidad Nacional de San Agustín.
29. Sarria, M.D.D., *Sistema de recomendación web basado en la actividad de los usuarios de la Universidad Nacional de Colombia*, in *Ingeniería de Sistemas e Industrial*. 2012, Universidad Nacional de Colombia.
30. Vázquez, F.I., *Métodos de agrupamiento en el cálculo de distancias entre usuarios de recomendación basado en algoritmos de filtrado colaborativo*, in *Departamento de teoría de la señal y telecomunicaciones*. 2011, Universidad Carlos III de Madrid.
31. Ghobar, E.W., *Un sistema de recomendación basado en perfiles generados por agrupamiento y asociaciones*, in *Departamento de Sistemas Informáticos y computación*. 2017, Universidad politécnica de Valéncia.
32. Ochoa, L.L. and K.R. Paredes, *Estudio Comparativo de Técnicas no Supervisadas de Minería de Datos para Segmentación de Alumnos.*
33. Cambronero, C.G. and I.G. Moreno, *Algoritmos de aprendizaje: Knn & Kmeans.* 2006.
34. Pascual, D., F. Pla, and S. Sánchez, *Algoritmos de agrupamiento.* 2007.
35. Vermurugan, T. and T. Santhanam, *Computational complexity between K-mean and K-medoids clustering algorithms for normal and uniform distributions of data points.* 2010.
36. González, D.P., *Algoritmos de agrupamiento basados en densidad y validación de clusters*, in *Departament de llenguajes y sistemas informátics*. 2010, Universitat Jaume I.
37. *International Conference on Information Security & Privacy* 11-12 December 2015.
38. Garre, M., J.J. Cuadrado, and M.Á. Sicilia, *1Dept. de Ciencias de la Computación ETS Ingeniería Informática Universidad de Alcalá*
39. Goodin, A.D., *La evolución del aprendizaje: más allá de las redes neuronales.* 2013.
40. Longoni, M.G., et al., *Modelos de redes neuronales perceptron multicapa y base radial para la predicción del rendimiento académico de alumnos universitarios*. 2010, Universidad Nacional del Nordeste.
41. Nacelle, A., *Redes neuronales artificiales*, in *Las redes neuronales de la biología a los algoritmos de clasificación*. 2009.
42. Lobo, C. and J. Fernando, *Slideshare.* 2015.
43. Sánchez, T.R., *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015: p. 16.
44. Oracle. *Java + Alice: Creative, animate, play, learn!* 2019; Available from: https://www.java.com.
45. Oracle. *What is java?* 2019; Available from: https://www.java.com.
46. NetBeans. *NetBeans IDE 8.2 Release Notes*. 2016; Available from: https://netbeans.org.
47. PostgreSQL. *Características, limitaciones y ventajas*. 2018 17/01/2018]; Available from: https://www.postgresql.org/.
48. Paradigm, V. *Visual Paradigm 8.0 (formerly VP-UML 8.0) Released*. 2019; Available from: https://www.visual-paradigm.com/cn/aboutus/newsreleases/vpuml80.jsp.
49. López, P., *Herramienta CASE Visual Paradigm*. 2017, Universidad de Cantabria.
50. Oracle. *Información general sobre JavaFX* 2019; Available from: https://www.java.com/es/download/faq/javafx.xml.
51. GLUON, *Scene Builder.* 2019.
52. Oracle, *JavaFX Scene Builder.* 2019.
53. Hilera, J.R. and D. Palomar, *Modelado de procesos de enseñanza-aprendizaje reutilizables con XML, UML e IMS-LD.* 2009.
54. Peri, A.A., *SIDRA: XML en la gestión y explotación de la documentación jurídica.* 2009.

55. Azcuy, R.A., *Módulo para la configuración y monitorización de réplicas con la herramienta SysmmetricDS para la arquitectura Xalix.* 2017, Universidad de las Ciencias Informáticas.
56. Sommerville, I., *Ingeniería de software*. Novena Edición ed. 2011. 792.
57. Rumbaugh, J., I. Jacobson, and G. Booch, *El lenguaje unificado del modelado. Manual de referencia.* 2000.
58. Larman, C., *UML y patrones.* 2004.
59. Larman, C., *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado*. 2003.
60. Usaola, M.P., *Patrones GRASP.* 2018: p. 18.
61. *Ingeniería de software II.* 2011: p. 17.
62. Gamma, E., et al., *Design Patterns: Elements of reusable object oriented programming*. 1995.
63. Guerrero, C.A., J.M. Suárez, and L.E. Gutiérrez, *Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos.* Información Tecnológica, 2013. **Vol. 24**: p. 103-114.
64. Tituaña, C. and G. Dario, *Análisis,diseño e implementación de una aplicación web de control de bodegas para la empresa NOVAMISIC, mediante la utilización de la plataforma Java Enterprise Edition JEE7 Web*. 2016.
65. Ronzalen, E.J.V., *Convenciones de codificación para el lenguaje de programación JAVA.* 2014.
66. Madruga, Y.C., *Módulo para la gestión de riesgos, recursos materiales, interesados y foros virtuales en el SGPE*. 2017, Universidad de las Ciencias Informáticas.
67. Peño, J.M.S. *Pruebas de software. Fundamentos y técnicas*. 2015.
68. Huaraca, A.G.V. *Pruebas de sistemas y Pruebas de aceptación*. 2013; Available from: http://academica-e.unavarra.es/handle/2454/4523.
69. Chiu, C.C., *Las pruebas en el desarrollo de software*. 2015, Universidad Nacional Autónoma de México. p. 46.
70. Peño, J.M.S., *Pruebas de Software. Fundamentos y Técnicas*. 2015, Universidad Politécnica de Madrid. p. Pág. 132.
71. Juristo, N., A.M. Moreno, and S. Vegas, *Técnicas de evaluación del software.* 2005.