Universidad de las Ciencias Informáticas

Facultad 4

Módulo recomendador de libros para la plataforma de recursos Félix Varela

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

Autor:

Yuniel Lavin Gé

Tutores:

MSc. Karenia Donatien Goliath

Ing. Yenima Hernández Orozco

Ing. Yaritza Bárbara González Ramírez

La Habana, Junio 2017

"Año 59 de la Revolución"

Declaro ser el autor del presente trabajo de diploma y concedo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

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

Autor

_____

Yuniel Lavin Gé

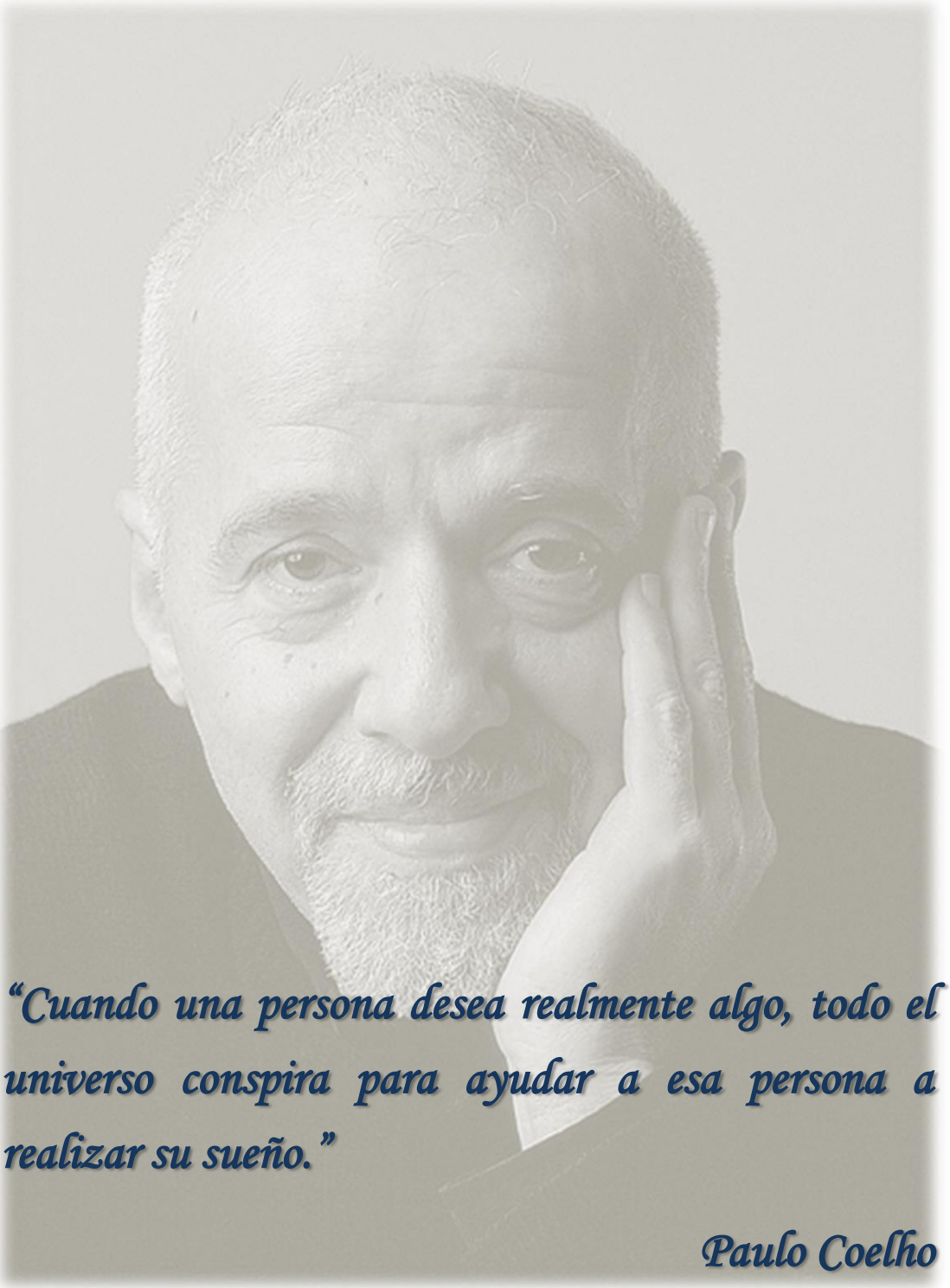
Tutores

_____

MSc. Karenia Donatien Goliath

_____ _____

Ing. Yenima Hernández Orozco Ing. Yaritza Bárbara González Ramírez


“Cuando una persona desea realmente algo, todo el universo conspira para ayudar a esa persona a realizar su sueño.”
Paulo Coelho

Dedicatoria

A mis padres por darme su apoyo incondicional en todo momento a pesar de las dificultades.

A mi hermano para que siga su camino como yo seguí el mío.

A mis abuelos paternos, que estarían muy contentos con mi graduación.

Agradecimientos

Llegando a este punto final es hora de agradecer a todos aquellos que me apoyaron y formaron parte de estos 17 años de estudio y preparación, principalmente estos últimos 5 años. Esta universidad me ha dado los mejores momentos de estudiantes y la oportunidad de conocer a personas muy importantes para mí, a todas las personas que aportaron su granito de arena, esto va para ustedes.

Primeramente agradecerles a mis padres por todo el amor y el apoyo que me dieron en toda esta etapa de mi vida, por su dedicación y sus consejos, sin los cuales no habría llegado hasta aquí.

A mi familia, la más especial del mundo, por siempre estar al tanto de mí.

A todos mis amigos del primer año; a Carlos por ser más que un amigo, un hermano para mí.

A mi novia por siempre estar ahí para mí, en todo momento; sin ti esto no hubiera sido posible. Gracias por compartir tantos momentos, buenos y malos no importa, te amo con la vida.

A mis compañeros de brigada; a Oscar, Pinar, Adrián, Javier, Lennon, David, Maydalis, Rosmery, Regla, Leslie y todos aquellos que se fueron incorporando con el tiempo.

A Maydalis por siempre ayudarme con las dudas y preocuparse por mis resultados.

A mis amistades de la facultad 3, a Ernesto, Claudia, Adolfo, Yanet y demás.

A mis tutoras por su tiempo, esfuerzo, paciencia y dedicación.

A los especialistas del centro FORTES en especial a Elías, por ayudarme siempre que lo necesité.

A todos aquellos que de alguna forma influyeron en que hoy esté logrando este sueño, muchas gracias.

Resumen

El desarrollo alcanzado en la actualidad por las Tecnologías de la Información y las Comunicaciones (TIC) ha potenciado un gran impacto en todas las esferas de la sociedad, principalmente en el comercio electrónico. El presente trabajo de diploma desarrolla un módulo recomendador de libros para los usuarios que acceden a la plataforma de recursos Félix Varela, apoyándose en la técnica de filtrado colaborativo basado en usuario. Dicho módulo está enfocado a orientar a los usuarios con respecto a qué libros visitar, ayudando de esta forma a los usuarios a acceder a los materiales literarios según sus necesidades y preferencias de lectura, disminuyendo así el tiempo que emplean estos en la búsqueda de libros. Se realiza un estudio de las clasificaciones de los sistemas de recomendación, haciendo énfasis en sus principales características. Se describen los algoritmos y técnicas empleadas para realizar la recomendación, así como las herramientas y tecnologías utilizadas en la concepción de la misma. Se realiza el análisis y diseño necesario para la implementación según la metodología AUP-UCI y finalmente se valida la aplicación comprobando así el buen funcionamiento de la misma.

Palabras claves: libros, Filtrado colaborativo, módulo recomendador, correlación de Pearson, K-Nearest-Neighbor.

Índice

Introducción

El continuo desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha traído consigo la llegada del libro electrónico y el progreso vertiginoso de sus soportes tecnológicos, lo que ciertamente ha revolucionado la industria editorial internacionalmente y comienza a dejar sus huellas en el mercado. Las TIC cambiaron el entorno de trabajo de las bibliotecas y centros de documentación e información respecto al modo en que realizan los procesos y prestan sus servicios. En el ámbito de las aplicaciones web para la distribución de artículos literarios existe la gobernabilidad por parte de gigantes como Amazon y Google ebooks (1). En Cuba, el bloqueo económico impuesto por los Estados Unidos de América ha restringido el acceso a diversas esferas, una de ellas es el acceso a una gran parte de la información académica y científica, lo que impide un buen desarrollo educativo y profesional por parte de estudiantes de diferentes carreras universitarias.

Como parte de una alternativa a la anterior situación, en Cuba se han desarrollado plataformas web para facilitar el acceso a libros y a publicaciones científicas y literarias, ejemplo de esto es el portal de la Biblioteca Nacional José Martí disponible en el sitio[1], el portal de la Biblioteca Central de la Universidad de La Habana disponible en el sitio[2] y la plataforma Librería Virtual disponible en el sitio[3], esta última con soporte para compras online. Las bibliotecas por naturaleza cuentan con un vasto catálogo de artículos literarios, lo que para el cliente puede ser un problema a la hora de elegir un libro, para esto se crean las plataformas virtuales. Al contar con una gran cantidad de información, es necesario que se desarrolle un sistema que facilite los procesos de búsqueda para que los usuarios adquieran la mejor opción. Los mecanismos que facilitan estas operaciones son los Sistemas de Recomendación o Sistemas Recomendadores, estas son aplicaciones informáticas que ayudan a los usuarios a obtener los tópicos relevantes para ellos de un conjunto variado de información. La principal tarea de estos sistemas es la de ofrecerle al usuario una lista con las posibles opciones que puedan interesarle, ahorrándole al mismo un largo tiempo de búsqueda por la red. Donde más se evidencia la aplicación de estos sistemas es en campos como el comercio electrónico y páginas dedicadas al ocio, sin embargo diversas instituciones bibliotecarias se han sumado a este desarrollo tecnológico. Una de las instituciones que se suma a este proceso de informatización es la Empresa Editorial Poligráfica Félix Varela (EEPFV).

La EEPFV es la encargada de la realización del proceso de edición de libros académicos que se corresponden con los programas de estudio de las carreras del Sistema de Educación Superior

[1] http://www.bnjm.cu/
[2] http://www.uh.cu/
[3] http://www.libreriavirtual.cu/

cubano. Entre los servicios brindados por la editorial se encuentra la publicación y promoción de literatura académica y científica de una amplia variedad temática. Dichas publicaciones son de referencia para estudiantes y profesores cubanos y de otras naciones por lo que constituyen un vasto espacio de consulta y localización de libros y otros recursos educativos.

En la Universidad de las Ciencias Informáticas (UCI) radica el Centro de Tecnologías para la Formación (FORTES) perteneciente a la Facultad 4, en el mismo se lleva a cabo el proyecto Desarrollo de la plataforma de recursos para la Editorial Félix Varela, dicho proyecto tiene como objetivo desarrollar una plataforma para la gestión, publicación y promoción online de libros y otros recursos educativos utilizando tecnologías libres. La editorial posee un amplio catálogo de libros que se les brinda a todos los usuarios que acceden a la plataforma, lo cual genera un gran cúmulo de información. El alto flujo de esta, unido a la alta frecuencia de publicaciones puede provocar que los usuarios sean desbordados de información y se sientan incapaces de seleccionar el contenido adecuado. Además, aunque la aplicación posea una buena estructura que facilite el acceso a la información, la navegación por el sistema para encontrar lo que les interesa requiere de un tiempo que en ocasiones no poseen, provocando de esta forma el abandono por parte de los usuarios a realizar consultas de los libros de su preferencia.

A partir de la problemática descrita surge como **problema a resolver** en la investigación la siguiente interrogante: ¿Cómo reducir el tiempo de búsqueda de libros en la plataforma de recursos de la Editorial Universitaria Félix Varela, teniendo en cuenta las preferencias de los usuarios según su interacción con la aplicación?

Para lo cual se plantea como **objetivo general:** Desarrollar un módulo de recomendación que reduzca el tiempo de búsqueda de libros, teniendo en cuenta las preferencias de los usuarios que interactúan con la plataforma de recursos de la Editorial Universitaria Félix Varela.

Definiéndose como **objeto de estudio**: Los procesos de recomendación en las plataformas web.
El **campo de acción** se centra en el proceso de recomendación de libros en la plataforma de recursos de la Editorial Félix Varela.

A partir del objetivo general definido se derivan los siguientes **objetivos específicos:**

- Construir el marco teórico referencial mediante la consulta, extracción y recopilación de la información relevante sobre el problema a resolver.

- Desarrollar el análisis y diseño del módulo de recomendación de libros para la plataforma de recursos de la Editorial Félix Varela.
- Implementar el módulo de recomendación de libros para la plataforma de recursos de la Editorial Félix Varela.
- Validar la propuesta de solución verificando que cumpla con los requerimientos establecidos.
- Realizar la validación experimental de los resultados mediante la puesta en práctica de un cuasiexperimento, usando el diseño con posprueba únicamente y grupo de control.

Esperándose como **resultado de la investigación**: Módulo de recomendación de libros para la plataforma de recursos Félix Varela, el cual permitirá a los usuarios reducir el tiempo de búsqueda de los libros relevantes para ellos, teniendo en cuenta sus preferencias y su interacción con el sistema.

Para la realización de esta investigación se hace uso de los métodos teóricos y empíricos, los cuales permitieron develar la parte de la ciencia que está siendo objeto de estudio. Entre los primeros se emplean:

Métodos teóricos

- **Histórico-lógico:** Utilizado al tener en cuenta la caracterización de la evolución histórica de los sistemas de recomendación como herramientas bases para concebir el sistema actual.
- **Analítico-sintético:** El empleo de este método se evidencia cuando se realiza un análisis de toda la teoría y documentación, que permiten la extracción de los elementos fundamentales relacionados con la recomendación de libros.
- **Modelación:** Es el método mediante el cual se crea una representación de todos los datos esenciales referentes al proceso de recomendación de libros en la plataforma Félix Varela.

El uso de los **métodos empíricos**, por otro lado, conllevó a una serie de procedimientos prácticos, permitiendo revelar las características fundamentales a cumplir por el sistema.
Se empleó:
Entrevista al cliente y especialista en el tema para determinar las funcionalidades a implementar para desarrollar un módulo que recomiende libros en la plataforma Félix Varela.

Para dar cumplimiento a los objetivos propuestos, se definen las siguientes **tareas**:

1. Elaboración del estado del arte de los Sistemas Recomendadores y los principales conceptos y elementos teóricos del tema a tratar.

2. Estudio y selección de las técnicas, herramientas y metodologías a emplear en el desarrollo de la solución.
3. Redacción del diseño teórico metodológico de la investigación.
4. Descripción de la propuesta de solución, los actores que interactuarán con el sistema, así como de los requerimientos funcionales y no funcionales a implementar en la migración.
5. Análisis y diseño de la propuesta de solución mediante la confección de los artefactos definidos en la metodología de desarrollo seleccionada.
6. Estudio de patrones arquitectónicos y de diseño a emplear en el desarrollo del software.
7. Implementar el módulo de recomendación de libros para la plataforma de recursos de la Editorial Félix Varela, teniendo en cuenta las preferencias de los usuarios según su interacción con la aplicación.
8. Estudio sobre los niveles y métodos de prueba que se pueden aplicar para la validación del software.
9. Validación funcional de la solución propuesta, mediante las pruebas diseñadas y documentación de los resultados.
10. Validación experimental de los resultados mediante la puesta en práctica de un cuasiexperimento, usando el diseño con posprueba únicamente y grupo de control.

La presente investigación está organizada en tres capítulos, de la siguiente forma:

Capítulo 1- Fundamentación teórica: Se realiza un análisis de los principales conceptos relacionados con el objeto de estudio, así como un análisis del estado del arte de los Sistemas Recomendadores en apoyo a las plataformas web y en sentido general. También se realiza un estudio y selección de las distintas herramientas y tecnologías a utilizar en el desarrollo del sistema propuesto.

Capítulo 2- Descripción de la propuesta de solución: Se presenta el diseño de la propuesta de solución al problema planteado, definiéndose los servicios que brindará el módulo recomendador, las funcionalidades que debe cumplir y el diseño del mismo.

Capítulo 3- Implementación y pruebas: Abarca todo lo relacionado con la implementación del sistema y el proceso de pruebas utilizado. Se implementan todas las funcionalidades identificadas, logrando un sistema que permita cumplir el objetivo general de los estudios que se presentan. Se detallan también las pruebas realizadas que permitan validar los requerimientos funcionales de la aplicación.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En el contexto de las aplicaciones web, el problema de la sobrecarga de información y lo que trae consigo, ha sido tratado adaptando diferentes medidas para darle solución a estos problemas, una de estas medidas es el desarrollo de Sistemas Recomendadores (SR).

En este capítulo se lleva a cabo un análisis de los principales conceptos relacionados con los sistemas de recomendación y sus características. Además se realiza un estudio de la aplicación de los SR tanto nacional como internacional, también se ve reflejado un análisis de las herramientas y tecnologías a utilizar, dentro de estas la metodología a utilizar, los lenguajes a emplear en la solución que se propone, las herramientas de desarrollo y modelado, y la arquitectura de software.

1.2 Definición de Sistemas Recomendadores

Los SR son agentes de software que obtienen los intereses y preferencias de los consumidores individuales [...] y hacen recomendaciones en consecuencia. Tienen el potencial para apoyar y mejorar la calidad de las decisiones de los consumidores mientras que estos realizan la búsqueda y selección de productos en línea. (2)

Los SR son herramientas cuyo objetivo es asistir a los usuarios en sus procesos de búsqueda de información, ayudando a filtrar los ítems[4] de información recuperados, usando recomendaciones propuestas sobre esos ítems. Dichas recomendaciones se generan a partir de las opiniones proporcionadas por otros usuarios sobre ciertos ítems, tales como documentos, libros e informes en búsquedas previas o bien a partir de las preferencias del usuario objeto de la recomendación. (3)

Los (SR) son herramientas de software las cuales proveen sugerencias de productos para el usuario. Las sugerencias se refieren a diversos procesos de toma de decisiones, como qué artículos comprar, qué música escuchar, o qué noticias en línea leer. Estos sistemas normalmente se enfocan en un tipo específico de producto y en consecuencia de diseño, su interfaz gráfica de usuario, y la técnica de recomendación básica utilizada para generar las recomendaciones son todas personalizadas para proporcionar sugerencias útiles y eficaces para ese tipo de elemento (4).

[4] Elemento con características que lo identifican. En la presente investigación el concepto ítem será un referente a libro.

A continuación se muestra una imagen que representa el funcionamiento de un Sistema Recomendador:

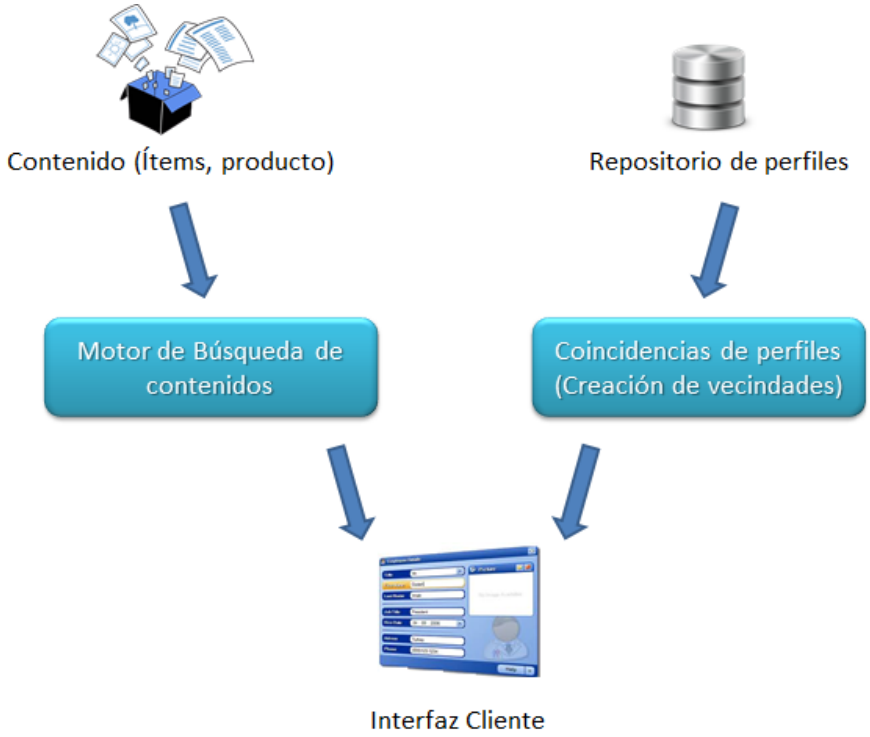


Figura 1: Esquema del funcionamiento de un Sistema Recomendador propuesto en **(5)**.

1.3 Problema de la recomendación

El principio de funcionamiento de los sistemas recomendadores se puede resumir en el problema de la recomendación, formulándose de la siguiente forma:

Al tener un conjunto U de usuarios y otro conjunto I de todos aquellos ítems que pudieran ser recomendados, siendo ambos conjuntos mayormente vastos, la recomendación sería aquella función que puede evaluar cuan útil puede ser el ítem $i \in I$ para el usuario $u \in U$, deseándose para cada usuario aquel ítem de mayor utilidad para el mismo o sea maximizar dicha función. Más formalmente: (6)

Sea U el conjunto de usuarios de la aplicación, I el conjunto de ítems, F la función evaluadora
Con $i \in I \; \forall \; u \in U$:

$$i_u = argmax_{i \in \mathrm{I}} (u, \mathrm{i})$$

Ecuación 1.1: Representación matemática del problema de la recomendación (6).

Con esta definición la diferencia entre los problemas de recomendación se centra principalmente en la función de utilidad y en el contexto del problema.

1.4 Modelo del proceso de recomendación

En el contexto de los sistemas de recomendación, los componentes que ocupan un papel central son el buscador de recomendaciones, el proveedor de preferencias y el sistema recomendador en función. El buscador de recomendaciones (por lo generar un usuario) puede solicitar sugerencias al sistema recomendador, o este puede automáticamente ofrecerlas sin una solicitud previa a partir de los mecanismos que se definen es su concepción. Para la construcción de las recomendaciones, el sistema puede capturar de manera directa las preferencias específicas de los usuarios, o puede inferirlas a través de preguntas en forma de formulario o el individuo calificar el ítem. Una vez recopilados todos los datos sobre los usuarios, el sistema está listo para recomendar los ítems que el usuario probablemente prefiera, basándose en su perfil, el perfil de otros usuarios, los datos del proveedor de las posibles preferencias, es decir, se conforma la vecindad a partir de las similitudes entre individuos, siendo utilizada esta vecindad para seleccionar ítems del universo de alternativas (7).

1.5 Ventajas e inconvenientes de los Sistemas Recomendadores

1.5.1 Ventajas

Son varias las razones por las cuales los proveedores de servicios pueden estar interesados en incursionar en este tipo de tecnología, las principales son (4):

- **Incremento del número de productos vendidos:** La capacidad de vender productos adicionales a los vendidos sin ninguna recomendación. Esto es posible porque los productos recomendados generalmente satisfacen lo que el usuario necesita y quiere.
- **Mayor venta de productos diversos:** Los SR permiten a los usuarios seleccionar productos que pueden ser difíciles de encontrar sin una recomendación específica.
- **Incremento de la satisfacción del usuario:** Un buen diseño de un SR puede mejorar la experiencia del usuario con el sitio web o la aplicación. El usuario encontrará las recomendaciones interesantes y relevantes. Las recomendaciones acertadas y una interfaz intuitiva harán que el usuario tenga una experiencia satisfactoria.

1.5.2 Inconvenientes de los SR

En los sistemas recomendadores existen problemas que se deben considerar para su diseño, Macmanus plantea algunos indicios de los problemas que deben superar las organizaciones para construir sistemas recomendadores eficaces (8):

- **Carencia de información:** Para realizar una correcta recomendación se necesita gran información, ya sea por parte de los usuarios o de los ítems. Los sistemas recomendadores tienen mayor exactitud de sus predicciones cuando cuentan con mayor cantidad de información disponible.
- **Información cambiante:** El problema de este tipo radica cuando un determinado ítem está de moda y recibe gran cantidad de valoraciones. Cuando este período termina, los usuarios no desean que se les recomiende más; sin embargo, el sistema de recomendación los sigue recomendando pues tiene un rating elevado.
- **Ítems impredecibles:** En ocasiones existen ítems difíciles de recomendar pues a los usuarios no les resulta de interés.
- **Cambio de preferencias del usuario:** En ocasiones los usuarios buscan recomendaciones para ellos mismos, pero pueden buscar productos para otros usuarios con preferencias y gustos diferentes a los suyos.

1.6 Clasificación de los sistemas recomendadores

A partir de la bibliografía consultada y considerando los aspectos establecidos anteriormente, los sistemas recomendadores pueden recibir diferentes clasificaciones dependiendo del tipo de información que utilizan para realizar sus recomendaciones. La clasificación más aceptada y reconocida por los principales autores en el tema son los que a continuación se exponen:

Sistemas Recomendadores basados en filtrado colaborativo:

La implementación original más simple de este tipo de sistema es la de recomendar al usuario los tópicos que han elegido otros usuarios que comparten gustos similares. Aquí la similitud en el gusto de dos usuarios se calcula en base a las coincidencias encontradas en las calificaciones que dieron ambos a distintos tópicos. Es por esto que algunos autores se refieren al filtrado colaborativo como “correlación persona-persona”. Estos sistemas de recomendación presentan elementos que han gustado a otros usuarios con gustos similares, con este propósito, calculan la similitud entre usuarios. En estos sistemas el usuario debe realizar una evaluación previa sobre algunos elementos. De esta forma se va formando el perfil del usuario. (9)

Para cada usuario se crea un conjunto de "vecinos cercanos", usuarios cuyas evaluaciones anteriores tienen grandes semejanzas a las del usuario en cuestión. Los resultados para los elementos no calificados se predicen en base a la combinación de puntos (scores) conocidos de los vecinos cercanos (10). A continuación se explican los tipos de filtrado de información según esta técnica de recomendación: (9)

- **Filtrado basado en usuario:** Sugiere a cada usuario activo aquellos productos que han interesado a sus vecinos. Para formar este vecindario, la estrategia considera que dos usuarios tienen preferencias similares si han clasificado los mismos productos en sus perfiles y les han asignado índices de interés parecidos.
- **Filtrado basado en ítem:** Mediante este método, un producto es recomendado a un usuario activo si es similar a los definidos en su perfil personal. En este caso, se considera que dos productos son similares (o vecinos) si los usuarios que han clasificado uno de ellos tienden a clasificar el otro, asignándole índices de interés parecidos.

Sistemas Recomendadores basados en el conocimiento:

Según el Lic. Enrique Antonio Cingolani, estos sistemas recomiendan los tópicos basados en conocimiento específico de dominio. Interpretan como ciertas características de un tópico cumplen con los requerimientos y preferencias del usuario y en qué medida este tópico puede serle útil. Muchos de estos sistemas están basados en casos. Utilizan una función de similitud con las recomendaciones (solución del problema). El valor que toma esta función se interpreta directamente como la utilidad que tiene la recomendación para el usuario. (10)

Un tipo especial de sistemas recomendadores basados en conocimiento, son los sistemas basados en restricciones. Ambos tipos son similares desde el punto de vista del conocimiento utilizado. La diferencia más grande entre sistemas recomendadores basados en casos y basados en restricciones es que los primeros usan métricas de similitud, en tanto que los segundos utilizan por lo general bases de conocimientos que contienen reglas explícitas que indican como relacionar los requerimientos del usuario con las características del tópico.

Los SR basados en el conocimiento tienden a trabajar mejor que otros al principio, pero si no están bien equipados con componentes de aprendizaje son pronto superados por otros que utilizan métodos que explotan las interacciones de las personas con las computadoras (como los de filtrado colaborativo, en comunidades con número creciente de usuarios).

Sistemas Recomendadores basados en contenido:

Los sistemas de recomendación basados en contenido definen los ítems por sus características y recomiendan al usuario ítems con características similares a los que ha puntuado favorablemente en el pasado. Así que la recomendación busca ítems similares al actual. Los sistemas de recomendación que usan ésta técnica sufren algunos problemas que también son presentes en los sistemas que utilizan técnicas basadas en un filtro colaborativo, como el problema de un nuevo usuario y el de dispersión de ratios. Pese a ello, la principal desventaja de ésta técnica es que los sistemas ofrecen ítems similares a los vistos con anterioridad, lo que significa que no introducen ninguna novedad. (11)

Sistemas de recomendación híbridos:

Según Helena Muños Escudero los sistemas híbridos adoptan un enfoque equilibrado, principalmente, entre un sistema colaborativo y basado en contenidos, con el fin de sacar el máximo partido a las ventajas de ambos, evitando las principales carencias. La mayoría de sistemas que usan la técnica de recomendación híbrida están compuestos por dos o más sistemas de recomendación, los resultados de los cuáles son combinados de distintas maneras para obtener un resultado más fiel.

Otra definición la expresa Fernando Andrés Peña y Ricardo Elías Riffo Carrillo:

Las limitaciones identificadas tanto en el filtrado demográfico, como en los métodos basados en contenido y el filtrado colaborativo, plantearon la necesidad de combinar varias de estas estrategias para así aunar sus ventajas y mitigar sus deficiencias. Bajo esta premisa, surgen los denominados sistemas híbridos.

Uno de los modelos que goza de mayor uso es el que combina el filtrado colaborativo y los métodos basados en contenido. La mayoría de las propuestas existentes adoptan un paradigma conocido en la literatura como collaboration via content, definido por Pazzani en 1999. Este autor propone calcular la similitud entre los usuarios del sistema utilizando tanto las descripciones semánticas de los productos definidos en sus perfiles (empleadas en los métodos basados en contenido), como los niveles de interés asignados a los mismos (considerados en el filtrado colaborativo).

1.7 Evolución de los Sistemas Recomendadores

Es evidente que los productos cambian, los usuarios también, así como sus predilecciones, poder adquisitivo, las modas, las preferencias sociales. Hay una gran variedad de factores y son necesarias nuevas ideas para obtener información valiosa, utilizarla con eficacia y que produzca con mucha frecuencia el resultado deseado en el cliente. No se trata únicamente de aumentar las ventas, sino de obtener un cliente satisfecho que vuelva.

Por estas y otras razones surgen iniciativas críticas y a la vez innovadoras en la forma de plantear los SR para satisfacer más adecuadamente las necesidades de los usuarios (12).

En el prólogo del libro “Recommender Systems” de Joseph A. Konstan, hace recapitulación de lo que él considera han sido las cuatro faces históricas de los sistemas de recomendación, las cuales son: (13)

1. Exploración orientada en sistemas.
2. Rápida comercialización (los retos de escala y valores).
3. Explosión de la investigación (los sistemas de recomendación se vuelven comunes).
4. Avanzado (recomendadores en contexto).

El término de sistema recomendador surge en la década de los 90, específicamente con el sistema PARC en 1992, el cual introdujo la idea de filtros colaborativos y mostró como anotaciones explícitas y datos de comportamiento implícito pueden ser obtenidos y utilizados en una base de datos para generar filtros personales. En 1995 el sistema Ringo en el Massachusetts Institute of Technology (MIT) mostró un acercamiento de filtros colaborativos que pueden ser distribuidos en una red y a su vez automáticamente sobre un servicio de álbumes de música y artistas, este sistema utilizaba técnicas de automatización (13).

1. Exploración orientada en sistemas

Esta fase fue caracterizada por la demostrada eficacia de los sistemas de recomendación generando gran emoción en el avance del campo, sistemas como Findme en 1996 que no solo fue basado en filtros colaborativos sino también emplearon conocimiento basado en el sistema. Uno de los eventos importantes que se realizaron en esta fase fue “The Collaborative Filtering Workshop” en Berkeley en marzo de 1996, en el cual se identificó un nuevo problema que rápidamente fue conocido como sistemas de recomendación (13).

2. Rápida comercialización

Los SR surgieron durante un período de rápida expansión del internet donde aparecieron empresas como “Agents, Inc” y Net Perceptions, estas nuevas compañías empezaron a enfrentar problemas del mundo real y fue necesario mostrar predicciones adecuadas y no afectar la eficiencia de las páginas de esa era, estos sistemas tenían que manejar millones de usuarios y productos y cientos o miles de transacciones por segundo. Diversos algoritmos surgieron durante esta era como algoritmos de correlación basados en elementos y enfoques de reducción de dimensiones, también la investigación se enfocó en evaluar recomendados basados en listas de las n mejores recomendaciones. La investigación se enfrentó a problemas relacionados con calificaciones implícitas, nuevos usuarios, nuevos elementos, experiencia de usuario, confianza, explicaciones y transparencia (13).

3. Explosión de la investigación

Esta fase fue aproximadamente entre el año 2000 y 2005, muchas compañías dedicadas a sistemas de recomendación cerraron debido a la explosión de la burbuja del internet, unas cuantas que lograron generar herramientas que abarcan diversas áreas pudieron permanecer con un gran uso en comercio electrónico, ventas minoristas y conocimiento de administración. Al mismo tiempo la investigación en sistemas de recomendación se amplió con la inclusión de diversos enfoques de diversas disciplinas como la inteligencia artificial, recuperación de la información, minería de datos, seguridad, privacidad, negocio y mercadotecnia (13).

4. Avanzado

En 2006 MyStrands organizó Recommenders06 que es una escuela de verano sobre el presente y futuro de los sistemas de recomendación. En 2007 fue organizada la primera conferencia de sistemas de recomendación (ACM Recommender Systems Conference). En las cuales ha existido interés en ver las recomendaciones en su contexto, construcción de nuevas herramientas de investigación para fomentar los recomendadores en un entendimiento de como las personas interactúan en las organizaciones y negocios. Uno de los artículos más famosos de estas conferencias es “Collaborative Prediction and Rankings with Non-random Missing Data” en el cual se plantea un método para evaluar algoritmos de recomendación y el artículo más citado es “Evaluating collaborative filtering recommender systems” que explica como empatar la evaluación con las necesidades del usuario (13).

1.8 Algoritmos en los sistemas recomendaciones

Un algoritmo se puede definir como una secuencia de instrucciones que representan un modelo de solución para determinado tipo de problemas, o bien como un conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema (14). En el contexto de los sistemas recomendadores los algoritmos son técnicas que se basan en la relación del usuario **U** con el ítem **I** para resolver el problema de la recomendación. Las técnicas algorítmicas se pueden agrupar en dos grandes grupos que a continuación se explican (15).

- **Técnicas basadas en memoria:** emplean métricas de similitud para determinar el parecido entre una pareja de usuarios. Para ello calculan los ítems que han sido votados por ambos usuarios y comparan dichos votos para calcular la similitud.
- **Técnicas basadas en modelos:** utilizan la matriz de votaciones para crear un modelo a través del cual establecer el conjunto de usuarios similares al usuario activo. Algunos ejemplos de estos modelos son los clasificadores bayesianos, las redes neuronales, algoritmos genéticos,

sistemas borrosos y la técnica de descomposición matricial basada en la técnica matemática del SVD (Descomposición de valores singulares).

1.8.1 Técnicas algorítmicas basadas en memoria

Las técnicas algorítmicas basadas en memoria utilizan la Base de Datos completa de usuarios-ítems para generar las predicciones, el Sistema Recomendador usa una técnica estadística para hallar un conjunto de usuarios que tengan una historia de concordancia con el usuario objetivo (vecinos), una vez es formada una vecindad se utiliza un algoritmo que combine preferencias de los vecinos para producir una predicción y un ranking de los “N principales” para el usuario objeto (5).

La definición anterior se refleja en la siguiente relación: sea R una matriz de relación usuario-ítems la cual contiene la evaluación (Ratings) de cada ítem por el usuario. Se toman dos usuarios ui y uj que pertenecen a R y se aplican algunos de los siguientes métodos:

Algoritmo de vecinos más cercanos o k-nearest neighbor (KNN)

Los clasificadores basados en memoria funcionan almacenando registros de capacitación y utilizarlos para predecir la clase (vecindad) de casos que no se ven. Este clasificador memoriza todo el conjunto de entrenamiento y clasifica solamente si los atributos del nuevo registro coinciden con uno de los ejemplos de entrenamiento exactamente. A más elaborado, y mucho más popular, el clasificador basado en memoria es el clasificador vecino más cercano (KNN). Teniendo en cuenta un punto para clasificarse, el clasificador KNN encuentra los k puntos más cercanos (vecinos más cercanos) de los registros de capacitación o entrenamiento. A continuación, se asigna la etiqueta de clase de acuerdo a las etiquetas de clase de sus vecinos más cercanos. La idea subyacente es que si un registro cae en una vecindad en particular, donde una etiqueta de clase es predominante es porque el registro es probable que pertenezca a la misma clase (16). A continuación se muestra el funcionamiento del proceso de recomendación utilizando la técnica KNN (17).

La implementación de la técnica de los k-vecinos en un algoritmo produce cuatro pasos que permiten establecer el grado de similaridad entre varios usuarios (Figura 2):

1. **Cálculo de similaridad entre usuarios:** De acuerdo a una serie de técnicas se procede a calcular la similaridad entre diferentes usuarios.
2. **Calcular los K-Vecinos:** A partir de los datos obtenidos con las técnicas se obtienen los usuarios con mayor grado de similaridad.
3. **Calcular las predicciones de los ítems:** Teniendo en cuenta los usuarios, se procede a predecir el valor que el usuario activo podría ofrecer sobre artículos que aún no ha valorado.

4. **Realizar las Recomendaciones:** Tras realizar los diferentes cálculos se procede a seleccionar los N artículos con mayor puntaje que pueden ser considerados para el usuario activo.

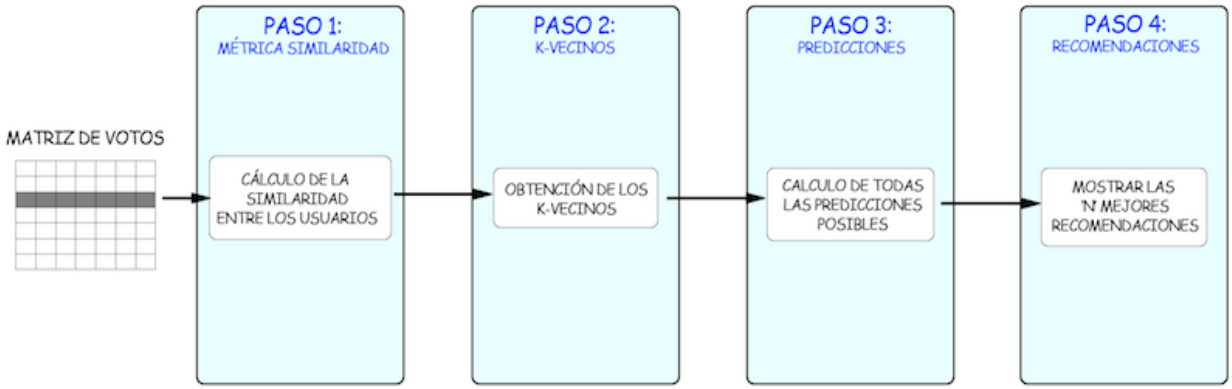


Figura 2: Pasos a seguir en el uso de la técnica KNN **(17)**.

Para la formación de relación de similitud entre todas las etiquetas (usuarios) y el usuario activo se definen algunas métricas:

- **Diferencia cuadrática Media (MSD)**

$$sim(x,y) = 1 - \frac{1}{\#B_{x,y}} \sum_{i \in I_u} \left(\frac{r_{x,i} - r_{y,i}}{max - min} \right)^2 \in [0,1]$$

Ecuación 1.2: Diferencia cuadrática media (MSD).

Siendo $B_{x,y}$ el número de películas que ambos usuarios han votado (y que tiene que ser necesariamente mayor que 0); siendo $r_{x,i}$ y $r_{y,i}$ los votos emitidos por los usuarios x e y respectivamente, y siendo max y min las notas máximas y mínimas que los usuarios han emitido (15).

- **Coeficiente de correlación de Pearson**

  El Coeficiente de correlación de Pearson es una métrica típica de similitud entre funciones de preferencias de usuarios o (menos frecuentemente) distancias de vectores o productos puntos. La fórmula da una aproximación de qué tan bien los vectores comparados (perfiles, ítems) coinciden en la escala desde cero (no similares) a uno (total coincidencia) ó -1 (total diferencia) (5).

$$\mathrm{sim}_{ik} = \mathrm{corr}_{ik} = \frac{\sum_{j=1}^{l}(r_{ij} - \bar{r}_i)(r_{kj} - \bar{r}_k)}{\sqrt{\sum_{j=1}^{l}(r_{ij} - \bar{r}_i)^2 \ \sum_{j=1}^{l}(r_{kj} - \bar{r}_k)^2}}$$

Ecuación 1.3: Correlación de Pearson (5).

i: Usuario activo.

k: Usuario al que se le desea calcular la correlación.

j: Artículo.

$\bar{r}$: Media aritmética del índice de preferencia de los artículos.

rij: Índice de preferencia del usuario i con el ítem j.

rjk: Índice de preferencia del usuario k con el ítem j.

- **Similitud basada en coseno**

  Esta técnica tradicionalmente vinculada al campo de la recuperación de información se utiliza en sistemas que representan las preferencias de los usuarios (y también los productos objetivos) mediante los vectores de características semánticas. El enfoque, propuesto por Salton en 1983, calcula la similitud entre dos vectores como el coseno del ángulo que forman, tal como se muestra en la Ecuación 1.4. Así, si ambos vectores son idénticos, la similitud toma el valor 1, mientras que si son completamente diferentes y, por tanto ortogonales, la similitud entre ambos se anula (9).

$$\mathrm{sim}_{\mathrm{Cos}}(A, B) = \frac{\overrightarrow{A} * \overrightarrow{B}}{\left|\overrightarrow{A}\right| * \left|\overrightarrow{B}\right|}$$

Ecuación 1.4: Similitud Basada en Coseno (9).

1.8.2 Técnicas algorítmicas basadas en modelo

Estos algoritmos primero modelan los votos de los usuarios. Tratan el problema como un problema de predicción estadística y calculan el valor esperado para cada ítem en función de los votos anteriores. Para ello se utilizan distintos algoritmos de aprendizaje: Clustering, Redes Neuronales como las Redes de Funciones de Base Radial (RBFN), Horting, Encasillamiento, Redes Bayesianas. En general, ante las consultas responden más rápido que los basados en memoria, pero necesitan de un proceso de aprendizaje intensivo (18).

- **Redes Bayesianas**

Una Red Bayesiana es un grafo acíclico en el que los nodos representan variables proposicionales y los arcos identifican dependencias causales entre ellos. Así, un arco entre dos nodos de la red significa que el nodo origen (también llamado nodo padre) tiene un impacto causal sobre el nodo destino (nodo hijo), o lo que es lo mismo, que este último depende del primero. Para poder cuantificar el efecto que los nodos padres tienen sobre un nodo específico de la red, se asocia a cada uno de éstos una tabla de probabilidades condicionales (calculadas mediante el teorema de Bayes), de forma que el valor de cada nodo es función de los valores de sus nodos padres. Además, los nodos hojas representan proposiciones cuyos valores son determinados por observación, a partir de los datos almacenados en un conjunto de entrenamiento del que se extrae la información representada en la red (19).

Las Redes Bayesianas son aplicables al campo de los sistemas de recomendación, son herramientas adecuadas tanto para modelar las preferencias de los usuarios, como para utilizarlas posteriormente durante el proceso de recomendación. El perfil de usuario resultante sería un grafo en el que los nodos y arcos identificarían los conceptos/atributos que interesan a los usuarios y las relaciones causales establecidas entre ellos, respectivamente. Para elaborar recomendaciones personalizadas, el clasificador automático decide si un producto objetivo es interesante para el usuario activo, basándose en las relaciones probabilísticas existentes entre los atributos de dicho producto y las preferencias representadas en su red Bayesiana (19).

- **Redes Neuronales**

  Las Redes Neuronales (RN) proporcionan una forma muy conveniente de representación del conocimiento en tareas de Recuperación de Información (RI), donde los nodos representan objetos del proceso de RI como palabras claves, autores y citaciones y los enlaces representan la asociación ponderada de estos (relevancia). Las propiedades de aprendizaje de las RN tipo retropropagación del error y las propiedades de búsqueda en paralelo de las Redes tipo Hopfield proporcionan mecanismos efectivos para identificar información relevante de ítems en bases de datos (5).

- **Horting**

  Es una técnica basada en grafos en la cual los nodos son los usuarios y las aristas entre nodos son indicadores de los grados de similitud entre dos usuarios. Las predicciones se producen al recorrer el grafo entre nodos cercanos y combinando las opiniones entre usuario cercanos. Esta técnica difiere de los algoritmos de vecindad más cercana, en la forma como el grafo puede ser recorrido por otros usuarios que no han valorado los ítems, luego esta técnica

explora las relaciones transitivas que los algoritmos de vecindad más cercana no tienen en cuenta (5).

1.9 Fundamentación de la técnica algorítmica seleccionada

El módulo Recomendador que se desea desarrollar estará apoyado en la técnica algorítmica basada en memoria, específicamente el algoritmo **vecinos más cercanos o k-nearest neighbor** ya que sus características se adaptan a la solución y es unos de los primeros algoritmos que se usan en los SR basados en filtrado colaborativo.

1.10 Estudio de sistemas similares

Como se explica anteriormente los sistemas recomendadores son los encargados de sugerir a un determinado usuario cierta información que se considere de interés para él. Su surgimiento y empleo en la educación está propiciado por la gran cantidad de software o plataformas educativas que han surgido como una medida de apoyo al proceso de enseñanza y aprendizaje.

En la presente investigación se realiza un estudio de varios Sistemas Recomendadores a nivel internacional y nacional con características similares a las del sistema que se pretende desarrollar:

1.10.1 Sistemas recomendadores a nivel internacional

- **Amazon**

  Es el sistema de recomendación utilizado por la tienda virtual Amazon y según se basa en un algoritmo de Filtrado colaborativo Item-to-Item, que a diferencias de los sistemas filtrado colaborativos tradicionales, el algoritmo de computación es escalable independientemente del número de clientes y del número de ítems en el catálogo de productos. Este realiza recomendaciones en tiempo real, escala a un conjunto de datos masivos, y genera recomendaciones de alta calidad. Como el nombre del algoritmo indica, este sistema de recomendación se enfoca en la búsqueda de productos similares y no en usuarios similares. Por cada compra y valoración explícita que los usuarios realizan a un producto el sistema busca los productos similares y añade este a los productos similares y luego realiza la recomendación de estos (20).

- **Filmaffinity**

  Filmaffinity es un sistema de recomendación de películas, documentales, cortometrajes, mediometrajes y series de televisión muy usado en la actualidad y que cuenta con millones de

usuarios. Se basa en el filtrado colaborativo. Como muestra la figura 3.1 una vez registrado un usuario deberá calificar con un valor del 1 al 10 el material cinematográfico para que el sistema le pueda ofrecer recomendaciones en base a esta valoración. El sistema crea las “almas gemelas” que son los usuarios con más coincidencias en las calificaciones. En cualquier momento se puede cambiar una calificación a un ítem. Además el sistema cuenta con un filtro para seleccionar las películas por género, año, país, etc. (21)

- **Tekstum**

  Es un sistema de recomendación de libros basado en Big Data e Inteligencia Artificial capaz de extraer las claves que motivan al lector a elegir una novela, ofreciéndole una “nube de sentimientos” que le oriente para elegir un siguiente título, es decir extrae las emociones de los usuarios y recomienda los libros a partir de estas. El sistema está dotado de un diccionario con cerca de 20.000 términos y un potente algoritmo de búsqueda que rastrea reseñas en blogs, webs especializadas, plataformas de venta y redes sociales, identificando las descripciones que los lectores hacen de las obras. La adaptabilidad que ofrece la herramienta, permite su aplicación tanto para enriquecer las plataformas de venta de las editoriales o agentes literarios, como para los lectores (22).

1.10.2 Sistemas recomendadores a nivel nacional

Cuba es un país donde el desarrollo de sistemas recomendadores, es aún incipiente. Sin embargo, la investigación acerca de estos sistemas que contribuyan al aumento de la calidad de las aplicaciones es un tema al cual los ingenieros, diseñadores y desarrolladores en general están prestando más atención. Además, en la bibliografía consultada los trabajos e investigaciones que se han encontrado provienen fundamentalmente de la UCI, sin embargo, pueden citarse algunas investigaciones que muestran resultados enfocados a implementaciones reales de sistemas recomendadores.

- En el año 2012 se desarrolló un sistema para la biblioteca virtual de la UCI disponible en el sitio[5], dicho sistema se encarga de la recomendación de artículos para catálogo en línea del portal. Este sistema recomendador tiene la característica que combina las técnicas basadas en contenido y las técnicas basadas en filtrado colaborativo, resultando así un SR híbrido que permite mitigar las limitaciones de las técnicas anteriores (23).

[5] http://biblioteca.uci.cu/

- En el año 2013 se creó un SR de ejercicios físicos para los estudiantes de la UCI. Su filosofía de recomendación está basada sistema de recomendación híbrido donde se orientan una serie de ejercicios a partir del estado inicial de condición física y así contribuir al mejoramiento de la condición física de los estudiantes de la comunidad universitaria. El sistema a desarrollar está apoyado en la técnica algorítmica basada en memoria, específicamente tomando como base el algoritmo KNN al cual se le realizó una modificación a partir de la idea general que plantea este algoritmo, lo cual permitió desarrollar un perfil algorítmico que se basa en la filosofía de recomendar al usuario un paquete de ejercicios basado en sus características y según sus necesidades y otro paquete apoyándose en las características de usuarios semejantes a él (7).

- Otro de los trabajos que pueden citarse orientados a la implementación de herramientas de recomendación en la UCI aplicados al campo de la enseñanza es en Recomendador basado en Filtrado Colaborativo para guiar el aprendizaje interactivo en el módulo Ejercicios de la colección El Navegante, que tiene como objetivo desarrollar un módulo recomendador de ejercicios basado en técnicas algorítmicas para el cálculo de la similitud entre usuarios , mejorando así la consolidación del conocimiento y el desarrollo educativo. Este módulo recomendador utiliza la técnica de filtrado colaborativo basado en usuario (24).

- En el año 2014 se desarrolló un módulo recomendador de libros para la tienda virtual del portal CubaLiteraria (25), el mismo se clasifica como sistema recomendador basado en contenido, enfocándose específicamente en el contenido generado por el perfil del usuario como método para filtrar las recomendaciones. Esta sistema se implementó con tecnologías como el CMS (Sistema Gestor de Contenido por sus siglas en inglés) Drupal y la metodología XP.

El estudio de estos sistemas fue provechoso, pues se identificaron características y funcionalidades comunes (creación del perfil de usuario, técnicas algorítmicas, tipo de sistema según el contexto) que sirven como base para el desarrollo del módulo recomendador propuesto. Además se obtuvo una visión de cómo manejar algunos de los procesos que se encuentran inmersos dentro de la estructura de los SR, facilitando el trabajo a la hora de tomar ideas para la construcción del mismo.

1.11 Herramientas y tecnologías a utilizar

Para llevar a cabo el ciclo de vida del desarrollo de software y garantizar que este proceso se realice de forma satisfactoria es necesario la aplicación de metodologías, herramientas y tecnologías. A

continuación, se realiza un estudio de las mismas, justificando así su selección a partir de las características que estas presentan.

1.11.1 Metodología de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Estas metodologías se dividen en dos grupos:

- **Metodologías Tradicionales:** ((RUP) Rational Unified Process), MSF (Microsoft Solution Framework),…) estas con mayor énfasis en la planificación y el control del proyecto.
- **Metodologías Ágiles:** ((XP) Extreme Programming, SCRUM,…) estas con la características de fácil adaptabilidad a cambios en los requisitos por parte del cliente y hará la entrega del proyecto más satisfactoria tanto para el cliente como para el equipo de desarrollo.

Actualmente el desarrollo de software dentro de la actividad productiva de la UCI se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles.

A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Las diferencias entre estas metodologías no radica únicamente en los productos de trabajos que proponen o en sus roles, sino en su forma de planificar el proyecto y realizar las estimaciones del tiempo. Factor determinante en la culminación exitosa de todo desarrollo de software, por lo que uno de los principales problemas detectados es que sin importar la metodología que se usa se está planificando con un único cronograma tipo, además de forzar el método de estimación definido en la Universidad y que responde en su gran mayoría a la metodología RUP (26).

Para lograr erradicar los problemas detectados, se propone una metodología para ser adaptada a lo que ya la Universidad ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles. La propuesta que se trae a continuación responde a una variación que se le realiza al Proceso Unificado Ágil.

El Proceso Unificado Ágil de Scott Ambler o Agile Unified Process (AUP) en inglés es una versión simplificada del Proceso Unificado de Rational (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo (26):

- Desarrollo Dirigido por Pruebas (test driven development -TDD en inglés).
- Modelado ágil.

- Gestión de Cambios ágil.
- Refactorización de Base de Datos para mejorar la productividad.

Fases de metodología AUP (26)**:**

1. **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto.
2. **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software.
3. **Cierre:** En esta se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.11.2 Lenguajes de programación del lado del cliente

En este apartado se realiza un análisis de los lenguajes de programación del lado del cliente a utilizar en la implementación de la solución, resaltando sus características y justificando así su selección.

HTML5 (HyperText Markup Language, versión 5)

HTML5 establece una serie de nuevos elementos y atributos que reflejan el uso típico de los sitios web modernos.

Actualmente en el desarrollo de la plataforma Félix Varela se utiliza HTML como lenguaje de etiquetas en su versión 5, ya que presenta nuevas ventajas a la hora de su implementación. Estas mejoras van enfocadas a una web con mejor rendimiento y apariencia ya que incorpora soporte con CSS3, mejora la velocidad de respuesta en la conexión con el servidor lo que proporciona una experiencia agradable al usuario (27); estas características son muy importantes a la hora de desarrollar un sistema como la plataforma Félix Varela.

JavaScript

JavaScript es un lenguaje de programación, al igual que PHP, si bien tiene diferencias importantes con éste, JavaScript se utiliza principalmente del lado del cliente (es decir, se ejecuta en el ordenador, no en el servidor) permitiendo crear efectos atractivos y dinámicos en las páginas web. Los navegadores modernos interpretan el código JavaScript integrado en las páginas web (28).

En la plataforma Félix Varela se utiliza el lenguaje JavaScript como herramienta para manejar los eventos como características de la web dinámica que se desea alcanzar, JavaScript además de ser un lenguaje ligero, es independiente de la plataforma en la que se trabaja, gratuito y cuenta con amplia documentación (29).

CSS3 (Cascading Style Sheets)

CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El CSS sirve para definir la estética de un sitio web en un documento externo y eso mismo permite que modificando ese documento (la hoja CSS) podamos cambiar la estética entera de un sitio web.

Se decide usar CSS en su última versión ya que en el proyecto Félix Varela se utiliza para aplicar estilos de última generación a la interfaz con la ayuda de otras herramientas. Esta versión de CSS incluye nuevas etiquetas que hacen más fácil el trabajo con multimedia mejorando así las deficiencias que presentaba CSS2, donde, para lograr estos efectos era necesario la aplicación de lenguajes externos como JavaScript (30).

1.11.3 Framework del lado del cliente

JQuery

Este es un framework de JavaScript que ofrece una infraestructura con la que se tendrá mayor facilidad para la creación de aplicaciones complejas del lado del cliente, además es gratuito y multiplataforma (31). La arquitectura a utilizar en el desarrollo de módulo recomendador integra a JQuery como unos de sus componentes por lo que su uso facilita el trabajo con la arquitectura.

Bootstrap v3.0

Es un framework para CSS utilizado en la plataforma Félix Varela para el diseño web ya que está basado en los últimos estándares de desarrollo web como son HTML5, CSS3, JavaScript/JQuery. Además de ser sencillo y ligero, es compatible con todos los navegadores habituales. Este framework cuenta con Responsive Web Design lo que permite visualizar el contenido en un amplio rango de dispositivos. (32)

1.11.4 Lenguaje de programación del lado del servidor

PHP 7

PHP es un lenguaje de código diseñado específicamente para la web y que puede ser incrustado en HTML. Lo que distingue a PHP de algo del lado del cliente como JavaScript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente (33). En la propuesta de solución se usará PHP en su versión 7 porque es la que se utiliza en el desarrollo de la plataforma Félix Varela. Esta versión incorpora nuevas mejoras a partir de las deficiencias de PHP 5 y sus complementos (PHP 5.6 la última). Algunas de sus novedades según (34) son:

- **Un nuevo Zend Engine:** Zend Engine incorpora un nuevo motor, lo que mejora el rendimiento del lenguaje.

- **El doble de velocidad:** Gracias al nuevo motor PHPNG (PHP New Generation), la velocidad aumenta considerablemente, con la nueva característica que incorpora, la compilación **Jit,** que hace que el código se compile cuando se está ejecutando en lugar de antes de la ejecución**.**
- **Facilita el manejo de errores.**
- **Incorporación de nuevos operadores.**
- **Soporte para sistemas Windows de 64 bits.**
- **Las clases anónimas:** Una clase anónima es una clase sin nombre, definida en la misma línea de código donde se crea el objeto de la clase**.** PHP 7 te permite utilizar clases anónimas, una práctica bien establecida en otros lenguajes orientados a objetos como C# y Java.
- **Hace limpieza:** El principal objetivo de PHP 7 es el de liberar espacio para permitir la mejora de rendimiento, por lo que era necesario deshacerse de muchas funcionalidades en desuso y APIs antiguas que ya no eran compatibles con la mayoría de los servidores**.**

1.11.5 PHP-ML (PHP Machine Learning)

PHP-ML es una librería para php que brinda funcionalidades para el trabajo con Aprendizaje Automático tales como Clasificación, Agrupamiento, Reprocesamiento y funciones matemáticas relacionadas con las tareas anteriores. PHP-ML trabaja bajo la licencia MIT y requiere una versión igual o superior a php v7.0 (35). Para el desarrollo del módulo recomendador se usa esta librería en cálculo del índice de correlación de Pearson de los usuarios a partir de la fórmula de Correlación de Pearson que brinda esta librería, también brinda dentro de la tarea de Clasificación el algoritmo KnearestNeighbors, pero al realizar un análisis de este se decide no utilizar este servicio ya que su funcionamiento no se adapta a los requerimientos de la propuesta de solución; por lo que se decide implementar una variante que se ajuste al problema.

1.11.6 Framework del lado del servidor

Symfony 2.7.16

Para el desarrollo del sistema recomendador propuesto se utilizará el framework Symfony, este es un marco de trabajo basado en PHP y en la arquitectura Modelo-Vista-Controlador utilizado también en el desarrollo de la plataforma Félix Varela. Además sigue la mayoría de las mejores prácticas de desarrollo web y patrones de diseño a partir de la arquitectura Modelo-Vista-Controlador.

Principales características:

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix-like estándares).
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de ORM (Doctrine 2, Propel), permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos y características como los espacios de nombres, de ahí que sea imprescindible PHP 5.3.
- Aunque utiliza MVC (Modelo Vista Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, adaptándose a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo.

1.11.7 Sistema gestor de base de datos (SGBD)

Según Juan Iruela, un Sistema Gestor de Bases de Datos es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible. En la actualidad, existen multitud de SGBD en la mayoría relacionales, lo más usados son los siguientes: (36)

- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL
- DB2

Para seguir un ciclo de desarrollo compatible con el del equipo de desarrollo de la plataforma Félix Varela se decide emplear el uso del gestor PostgreSQL en su versión 9.4 ya que es un gestor robusto

perfecto para almacenar grandes cantidades de datos, además de ser confiable y seguro ya que incorpora acceso encriptado vía SSL. (37)

1.11.8 Servidor web

Apache 2.4.7

Se escoge el servidor web apache para alojar de forma local el sistema recomendador propuesto ya que este es el que se encarga de soportar la plataforma Félix Varela, Apache es servidor libre y de código abierto, es uno de los servidores web más usados en el desarrollo de aplicaciónes en Internet. Entre las principales características se encuentran (38):

- Apache se ha concentrado en la escalabilidad, en la seguridad y en el rendimiento.
- Es un servidor web altamente configurable.
- Plataforma de servidores web de código fuente abierto.
- Trabaja con diversas tecnologías como PHP, mod_perl, servlets de Java, etc.

Existe compatibilidad entre el servidor web Apache y el framework Symfony por lo que es perfecto para el desarrollo de la propuesta de solución.

1.11.9 Lenguaje de modelado

El Lenguaje de Unificado de Modelado (por sus siglas en ingles **UML: Unified Modeling Language**) es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML ofrece una forma de modelar entes conceptuales como son los procesos de negocio y funciones de sistema, además de entes concretos como son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (39). Al ser utilizado en el desarrollo de la plataforma Félix Varela, se decide utilizar también en el modelamiento de los procesos de la presente propuesta de solución.

1.11.10 Entorno Integrado de Desarrollo (IDE)

Un Entorno de Desarrollo Integrado (IDE), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Los IDEs proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (40).

Según el sitio web Fergaciac disponible en (https://fergarciac.wordpress.com) un IDE debe tener las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda

NetBeans IDE

NetBeans IDE es un entorno de desarrollo integrado (IDE), modular, de base estándar (normalizado), escrito en el lenguaje de programación Java. NetBeans consiste en un IDE libre, de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación (41). NetBeans IDE 8.0.2 ofrece analizadores de código y editores para trabajar con las últimas tecnologías como Php v7.0. El IDE también tiene una gama de nuevas características que mejoran su soporte para Framework y Sistemas Gestores de Contenido (CMS); nuevas herramientas para HTML5 y mejoras a PHP (42).

PhpStorm 2016.2

PhpStorm es un IDE de programación desarrollado por JetBrains. Es uno de los entornos de programación más completos de la actualidad, permite editar código no sólo del lenguaje de programación php como lo indica su nombre (43). Este IDE tiene el inconveniente de que es un software propietario, por lo que es superado en este punto por otros sistemas libres enfocados al desarrollo web.

PhpStorm es perfecto para el trabajo con Symfony, Drupal, WordPress, Zend Framework, Laravel, Magento, Joomla, CakePHP, Yii, y otros frameworks. Actualmente las nuevas versiones proporcionan la mejor terminación de código, refactorings y prevención de errores en la marcha. Aprovecha al máximo las avanzadas tecnologías de front-end, como HTML5, CSS, Sass, Menos, Stylus,

CoffeeScript, TypeScript, Emmet y JavaScript, con refactorizaciones, depuración y pruebas de unidades disponibles. PhpStorm incluye en sí todas las características de WebStorm y el soporte completo para PHP y soporte para bases de datos como PostgreSQL, MySQL, SQLite y otros (44).

Este IDE es reconocido por su depurador visual de configuración cero, proporcionando una visión extraordinaria de lo que sucede en su aplicación en cada paso. Funciona con Xdebug y Zend Debugger, y puede usarse tanto local como remotamente. Unidad de pruebas con integración PHPUnit, BDD con Behat y Profiler también están disponibles (44).

Se decide utilizar la versión 2016.2 de PhpStorm ya que esta incluye nuevas mejoras como son:

- Mejor soporte para PHP 7.
- Inferencia de tipos, lo que mejora la inferencia de tipos de elementos como las matrices y los Iteradores, también aporta una comprensión aún más profunda del lenguaje PHP y ayuda a escribir menos para que se escriba su código.
- Un completado de código inteligente para facilitar el desarrollo.

1.11.11 Fundamentación del IDE de desarrollo a utilizar

De los dos IDEs analizados anteriormente se decide hacer uso del IDE NetBeans en su versión 8.0.2 ya que es un producto libre y gratuito el cual facilita el desarrollo de aplicaciones web con php, además de tener un potente Debugger integrado y con la característica que se adapta a nuestra solución propuesta pues cuenta con soporte para el framework Symfony.

1.11.12 Herramienta Case

Visual Paradigm

Visual Paradigm es una herramienta para el desarrollo de aplicaciones utilizando modelado UML, ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas. Ofrece confiabilidad y estabilidad en la desarrollo orientado a objetos durante todo el ciclo de construcción de software (45), además cuenta con herramientas para integrar con el Gestor de Base de Datos PostgreSQL; por estas razones se decide utilizar para el modelado de los procesos.

1.11.13 Arquitectura de software

Xalix es un marco tecnológico, basado en el framework de desarrollo Symfony 2, que se rige por pautas y permite la integración de componentes de forma organizada, que puedan ser reutilizadas en favor de satisfacer las necesidades del cliente para el cual se esté trabajando (46). Xalix, desde el

punto de vista del desarrollo de software, es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado.

Este marco de trabajo, incluye una base de tecnologías para el desarrollo y define una estructura y un mecanismo de integración para los componentes creados. Adopta una arquitectura basada en componentes, donde cada elemento puede desacoplarse y evolucionar independiente uno del otro y son esas partes las que conforman los productos genéricos una vez ensambladas (47).

1.12 Conclusiones del capítulo

En este capítulo apoyado en los métodos de la investigación científicos se realizó un estudio bibliográfico con el cual se construyó el marco conceptual sobre los sistemas recomendadores, se obtuvo el conocimiento necesario referente a los principales conceptos, características y antecedentes de los SR. Se determinaron las técnicas de recomendación según su clasificación decidiendo así hacer uso de la técnica basada en filtrado colaborativo porque es la más adecuada a la propuesta de solución. Se llevó a cabo un estudio de las herramientas y metodologías a utilizar definidas por el marco de trabajo Xalix.

Capítulo 2: Descripción de la propuesta de solución

2.1 Introducción

A partir de la metodología de desarrollo seleccionada, en su segunda fase AUP-UCI define como se ejecutan las actividades requeridas para desarrollar el proyecto. Durante el desarrollo se modela el dominio, se obtienen los requisitos, se elabora la arquitectura y el diseño, se implementa y se libera el producto. En el siguiente capítulo se expone la realización de las tareas previstas en la segunda fase la metodología AUP-UCI.

2.2 Usuarios relacionados con el módulo recomendador

La plataforma de Recursos Félix Varela posee un grupo de funcionalidades y servicios entre los que se encontrará el módulo recomendador de libros. La implementación de esta plataforma se realizó a través de módulos, a los que se tiene acceso dependiendo del rol y los permisos de los usuarios. Tanto los roles como los permisos determinan el nivel de acceso de cualquier usuario que interactúe con el sistema a las funciones del mismo. Un usuario es cualquier persona que se relacione con el sistema, así sea a través de la vinculación al desarrollo del mismo o que de una forma u otra va a interactuar con la aplicación, incluyendo aquellos que se encargan de mantener el sistema actualizado y funcionando. El módulo propuesto dispone de los siguientes roles:

Tabla 1: Usuarios relacionados con el módulo recomendador.

Usuario	Descripción
Autenticado	Todos los usuarios deben autenticarse para acceder a cualquier opción que así lo requiera. Puede acceder a todas las opciones, menos a las de administración.
Administrador	Son aquellos usuarios que tienen todos los permisos para administrar las funcionalidades del sistema. Gestionan los contenidos, los usuarios y todos los servicios que se brindan.
Anónimo	Es la persona que navega por el sistema sin haberse registrado aún, interactúa con este sin privilegios y tiene la posibilidad de visualizar las diferentes opciones que

	brinda el mismo.

2.2.1 Perfil de usuario

En informática se entiende por perfil de usuario, el conjunto de características o preferencias que la persona tiene sobre sus búsquedas de Internet o en los sitios web que frecuenta. Es con base a los datos que introduce a través de los diversos sitios, como se va conformando dicho perfil. Por ejemplo, al registrarse a un sitio web determinado e introducir datos como su edad, país, género, etcétera, se pueden ir agrupando esas características para saber qué le puede interesar a ese usuario (48).

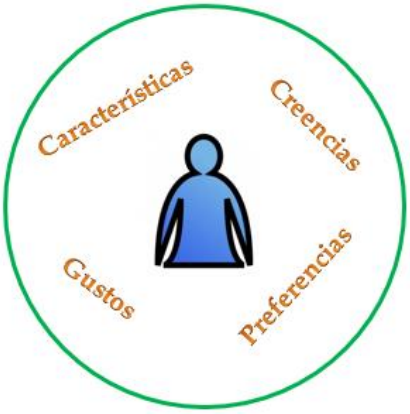


Figura 3: Un individuo con un conjunto de rasgos que lo caracterizan.

2.3 Modelo de dominio

Un modelo de dominio es una representación de conceptos en un dominio del problema. Un modelo conceptual explica (a sus creadores) los conceptos significativos en un área del problema analizado; es el artefacto más importante a crear durante el análisis orientada a objetos (49).

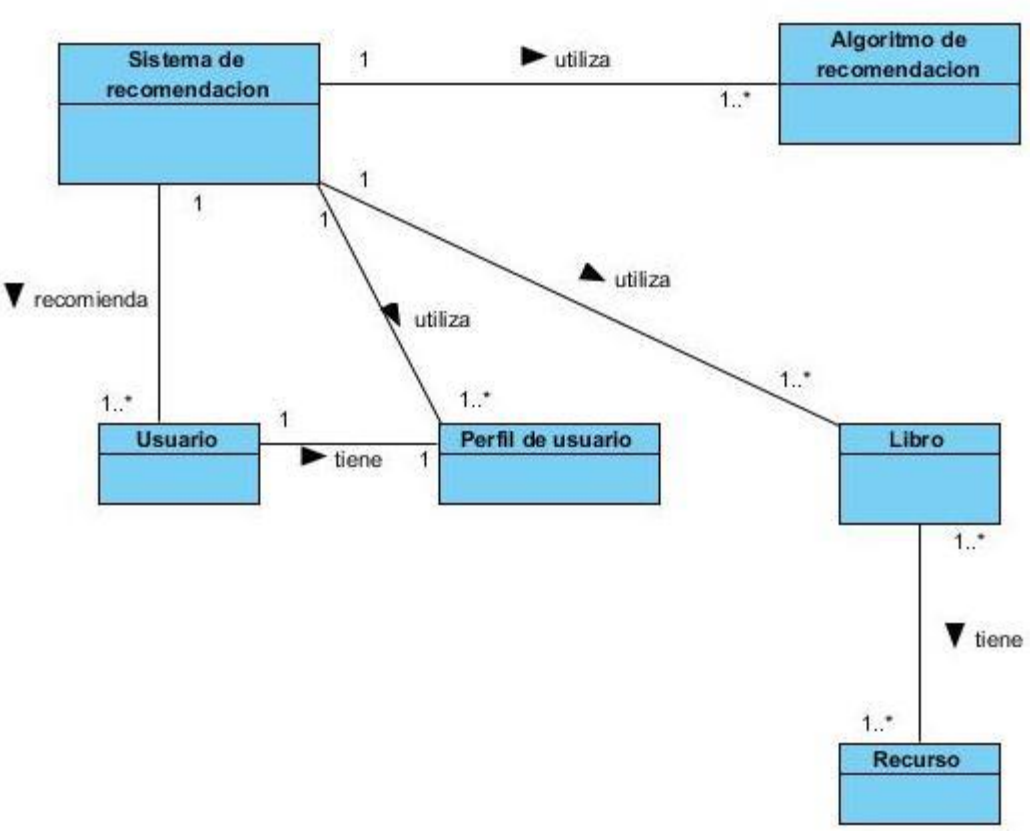


Figura 4: Modelo de dominio.

2.4 Fundamentación del tipo de sistema recomendador seleccionado

En este punto después de realizado un estudio bibliográfico respecto a los sistemas recomendadores se decidió darle solución al problema planteado en el capítulo anterior a partir de un módulo recomendador basado en filtrado colaborativo, con la particularidad de que este combinará las técnicas basadas en ítems y las técnicas basadas en usuario con el fin de evitar las desventajas que trae consigo la aplicación de esta última.

Desventajas de las técnicas basadas en usuario: (23)

- **Arranque frío:** Constituye uno de los problemas más difícil de resolver al inicio del despliegue de un sistema de recomendación, independientemente del objetivo con que se haya implementado. Este problema se debe a que en un principio se cuenta con muy pocas evaluaciones o consideraciones para realizar las recomendaciones.
- **Dispersión de los datos**: En mucho de los sistemas de recomendaciones que han sido comprobados en la práctica se tiene que, el número de evaluaciones reales que se dispone

acerca de un ítem en específico queda muy por debajo del número mínimo que se requiere para hacer las recomendaciones, haciéndose particularmente críticos para aquellos con preferencias alejadas de las de la mayoría. Esto implica que la predicción efectiva a partir de un número reducido de ejemplos juegue un papel clave para el éxito del sistema.

- **Sobre-especialización:** Está directamente asociada con los sistemas de recomendación basados en contenidos. Esto se debe cuando en estos sistemas solamente se recomiendan ítems que se correspondan a los que están almacenados en el perfil del usuario, obteniendo como resultado recomendaciones muy similares, puesto a que estas siempre se basan en la misma información y no tienen en cuenta las similitudes que pueden existir con otros usuarios, limitando de esta forma sugerencias de elementos similares que los mismos hayan evaluado o considerado.

2.5 Fases generales de los sistemas recomendadores de filtrado colaborativo

Para la realización de un sistema recomendador de filtrado colaborativo se deben tener en cuenta tres fases fundamentales por las que el mismo transita para determinar la recomendación final, para cada una existen técnicas ya estudiadas que ayudan en su construcción, la correcta selección y empleo de las mismas contribuirán en gran medida a que el recomendador logre los objetivos trazados (9).

1. **Selección de preferencias similares**

La primera fase determina las preferencias de todos los usuarios de la aplicación, mostrando un grado de correspondencia entre las similitudes de todos los usuarios del sistema y las del usuario activo. Para calcular las similitudes con el Coeficiente de Correlación de Pearson, primeramente es necesario tener un indicador o índice de preferencia de los artículos visitados por cada usuario.

Los ítems que se desean recomendar en la plataforma son los libros, cada libro se enfoca en un género o tema determinado, un usuario puede presentar problemas a la hora de seleccionar un artículo por el gran cúmulo de información, para resolver esto se necesita recomendarle los libros o artículos que le resulten de su interés atendiendo a su perfil y a sus preferencias. A partir de esto se calcula la similitud entre dos usuarios según los artículos a los que accede dentro de la plataforma. El índice de preferencia será dado entonces por las evaluaciones que realizan en cada artículo, teniendo en cuenta además el historial del usuario.

Este índice de preferencia es almacenado entonces en una matriz de usuario por artículo, representándose los artículos por columnas y los usuarios en las filas.

Ya reflejados todos los índices de preferencia de los usuarios en la matriz se pasa a calcular por cada usuario la correlación con el usuario activo. Definiéndose las variables de la fórmula de la correlación de Pearson.

$$sim_{ik} = corr_{ik} = \frac{\sum_{j=1}^{l}(r_{ij} - \bar{r}_i)(r_{kj} - \bar{r}_k)}{\sqrt{\sum_{j=1}^{l}(r_{ij} - \bar{r}_i)^2 \ \sum_{j=1}^{l}(r_{kj} - \bar{r}_k)^2}}$$

Ecuación 2.2 Fórmula de correlación de Pearson

El resultado ofrecido por la correlación de Pearson es un número en el intervalo [-1 ; 1], siendo este un indicador de cuán similar es el usuario con el usuario activo. Calculadas entonces todas las similitudes entre el usuario activo y el resto de los usuarios ya registrados en la plataforma, queda como resultado de esta fase una nueva lista donde se recoge de cada usuario la similitud que tiene con el usuario activo, la cual se utilizará entonces en la fase siguiente.

2. Creación del vecindario de un usuario

Luego del cálculo de las similitudes es necesario formar el vecindario del usuario activo, o sea, seleccionar un conjunto de usuarios con el objetivo de acotar las recomendaciones. Son tres las técnicas utilizadas tradicionalmente a la hora de determinar el tamaño del vecindario en los enfoques colaborativos:

Umbral: Se seleccionan sólo aquellos usuarios que superan un cierto umbral definido.

Los n mejores vecinos: Este método selecciona los n usuarios más parecidos al activo.

Centroide: Este enfoque selecciona el usuario más cercano al activo y calcula su Centroide. A continuación, incluye otros usuarios en el vecindario del activo, utilizando como criterio de selección la mínima distancia entre éstos y el Centroide.

De estas técnicas las más utilizadas y recomendadas son las dos primeras. Se ha decidido utilizar la técnica de los n mejores vecinos. Para el desarrollo de la solución propuesta se realizó un estudio de librerías que pudieran facilitar la implementación haciendo uso finalmente de la librería PHP-ML. Dentro de las funcionalidades que ofrece se encuentran las referentes a las tareas de la Minería de Datos[6] específicamente el Agrupamiento y la Clasificación. Uno de los algoritmos que implementa PHP-ML dentro de la Clasificación es el KNN, donde a partir de una métrica de distancia (Euclidiana,

[6] Proceso que intenta descubrir patrones de comportamiento en grandes volúmenes de conjuntos de datos.

Minkowski, Manhattan) y un umbral, calcula la distancia entre el usuario activo y el resto de los usuarios, los que queden con una distancia menor al umbral se definen como usuarios similares. A partir del funcionamiento de este algoritmo implementado por la librería se determina no hacer uso d este y concebir una nueva técnica que obtenga los vecinos más cercanos.

A partir del resultado arrojado por la fórmula de Correlación de Pearson se decide conformar una lista de usuarios con su índice de correlación asociado, dicha lista será ordenada descendentemente teniendo en cuenta el índice y acotándola a un máximo de diez usuarios (vecinos más cercanos).

3. Predicción basada en el vecindario creado

Finalmente el sistema debe predecir el nivel de interés del usuario activo en relación al producto objetivo. Para ello, los enfoques basados en usuario consideran el nivel de interés de los vecinos del usuario activo en relación al producto objetivo. También en este contexto, se han propuesto técnicas de diversa naturaleza, estas son:

- Recomendación de los productos más frecuentes.
- Recomendación basada en reglas de asociación.
- Media ponderada de clasificaciones.

De las técnicas anteriores las más recomendadas por la bibliografía para aplicar en los enfoques basados en usuario son la Recomendación de los productos más frecuentes y Media ponderada de clasificaciones.

Después del estudio realizado sobre las técnicas anteriores se decide escoger Media ponderada de clasificaciones ya que su funcionamiento depende de la aplicación de la fórmula de Correlación de Pearson para calcular la correlación de los usuarios activos con sus vecinos teniendo en cuenta los índices de preferencia por cada artículo y así calcular la media ponderada para posteriormente realizar la recomendación.

Media ponderada de clasificaciones: En las propuestas colaborativas basadas en usuario, un método alternativo para predecir el interés del usuario activo en relación a un determinado producto objetivo, consiste en promediar las clasificaciones que sus vecinos han asignado a dicho producto, empleando como pesos los valores de correlación entre sus respectivas preferencias. Esta idea aparece reflejada en la Figura 5, donde se muestra el nivel de interés $P_{U, i}$ que predice el enfoque para un usuario activo U y un producto objetivo i.

. A continuación se muestra la fórmula matemática de esta técnica (15):

$$p_{u,i} = \frac{\sum_{n \in G_{u,i}} sim(u,n) \cdot r_{n,i}}{\sum_{n \in G_{u,i}} sim(u,n)}$$

Figura 5: Ecuación de la media ponderada.

donde:

- u: usuario al que se le desea calcular la predicción.
- i: ítem.
- n: usuario correlacionado con u.
- $r_{n,i}$: evaluación de n al ítem i.
- sim(u;n): es la similitud (correlación) entre las preferencias del usuario u y n.

Después de aplicar esta técnica quedaría conformada una matriz de la siguiente forma:

$p_{u,i}$	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12
U1	1	1	4	3	2	1	3	3	4,5	4	•	1
U2	1	1	4	3,5	2	1	3	3	5	4	•	1
U3	2	•	5	2	1	•	1	4	•	1	2	1,5
U4	1,5	1	4,5	3	1,5	1	2	3	•	4	2	1
U5	2	•	5	•	1	•	•	4	•	1	2	2
U6	2	•	5	2	1	•	•	4	•	•	2	2

Figura 6: Matriz de predicciones.

Tomando por ejemplo la tabla anterior una vez que se han obtenido las predicciones sobre los ítems que el usuario no ha votado, solo queda realizar las recomendaciones y esto se hace ordenando las predicciones de mayor a menor y recomendar aquellas que mayor nota de predicción tengan. Por ejemplo, al usuario U1 el primer ítem a recomendar seria el ítem I9 ya que se ha predicho que el usuario votará dicho ítem con una nota de 4,5, luego se le recomendará el ítem I3 y así sucesivamente. Lo que se suele hacer en los sistemas de recomendación es recomendar un número determinado de ítems; 10 libros que corresponderán a las mejores predicciones. Para terminar este ejemplo, vamos a suponer que en nuestro sistema de recomendación vamos a recomendar solamente dos ítems, por tanto en la siguiente tabla vamos a mostrar que ítems son susceptibles de ser recomendados a cada usuario y cuales se les recomienda:

N=2	U1	U2	U3	U4	U5	U6
X_u	$\{I_3,I_4,I_6,I_9,I_{12}\}$	$\{I_1,I_2,I_5,I_9\}$	$\{I_3,I_5,I_7,I_{10},I_{11},I_{12}\}$	$\{I_1,I_5,I_6,I_8,I_{11}\}$	$\{I_8,I_{10}\}$	$\{I_1,I_{11}\}$
Z_u	$\{I_9,I_3\}$	$\{I_9,I_5\}$	$\{I_3,I_{11}\}$	$\{I_8,I_{11}\}$	$\{I_8,I_{10}\}$	$\{I_1,I_{11}\}$

Items susceptibles de ser recomendados (X_u) e items recomendados (Z_u) a los usuarios del filtrado colaborativo.

Figura 7: Recomendación de los n ítems de todos los posibles.

2.6 Requisitos de software

2.6.1 Requisitos funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (50)

A continuación se presentan los requisitos funcionales propuestos a cumplir:

Tabla 2: Descripción de los requisitos funcionales.

N°	Nombre	Descripción	Prioridad para el cliente	Complejidad	Referencias cruzadas
RF1	Crear perfil de usuario.	A partir de la navegación del usuario por la plataforma se almacenará en la base de datos las visitas y las evaluaciones de los libros.	Alta	Media	N/A
RF2	Calcular índice de correlación de Pearson.	A partir del usuario activo el sistema calcula el índice de correlación de este con los usuarios que han evaluado al menos un libro.	Alta	Alta	N/A
RF3	Establecer usuarios semejantes.	El sistema establece los usuarios más similares al usuario activo partir de su respectivo índice de correlación.	Alta	Media	N/A

RF4	Establecer ítems semejantes.	El sistema obtiene los libros similares a los visitados por el usuario activo teniendo en cuenta el autor, la categoría y la editorial.	Alta	Media	N/A
RF5	Calcular predicción de evaluación por usuario.	El sistema obtiene la predicción de evaluación que le daría el usuario activo a un libro n que no ha evaluado teniendo en cuenta los usuarios similares y la evaluación que estos le dieron a n.	Alta	Alta	N/A
RF6	Generar las recomendaciones basadas en ítem.	El sistema genera una lista con un máximo de 10 recomendaciones basadas en ítem.	Media	Media	N/A
RF7	Mostrar recomendaciones basadas en ítem.	El sistema muestra en la portada una lista de recomendaciones basadas en ítem.	Alta	Media	N/A
RF8	Generar las recomendaciones basadas en el usuario.	El sistema genera una lista con un máximo de 10 recomendaciones basadas en usuario.	Media	Media	N/A
RF9	Mostrar recomendaciones basadas en el usuario.	El sistema muestra en la portada una lista de recomendaciones basadas en usuario.	Alta	Media	N/A

RF10	Mostrar libros relacionados.	Cuando el usuario visita un libro aparece una pestaña “Libros relacionados” la cual muestra una lista de libros relacionados al visitado. Cuando se selecciona la opción “Ver” de la portada del libro se muestra una ventana con los principales datos del libro y una lista de libros relacionados.	Media	Alta	N/A
RF11	Mostrar libros recomendados por el sistema.	En la portada se muestra una lista de libros recomendados por el sistema donde se unen las recomendaciones basadas en ítem y las basadas en usuario.	Alta	Media	N/A

2.6.2 Requisitos 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 requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema. (50)

A continuación se presentan los requisitos no funcionales de la propuesta de solución:

Tabla 3: Descripción de requisitos no funcionales.

Atributo de Calidad	Portabilidad
Sub-atributos/Sub-características	Instabilidad
Objetivo	El sistema debe visualizarse y ejecutarse correctamente en un navegador web moderno, especialmente en Firefox (v20.x en adelante) y Google Chrome (v26.x en adelante), que son dos de los navegadores que mejor funcionan con HTML5 y CSS3.
Origen	Humano o cualquier sistema.
Artefacto	Todo el sistema

Entorno	Plataforma de recursos Félix Varela.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Despliegue de la aplicación	
Instalar la aplicación.	Sistema funcional, el cual se podrá acceder desde una PC cliente.
Medida de respuesta	
N/A	
Atributo de Calidad	Eficiencia
Sub-atributos/Sub-características	Utilización de recursos
Objetivo	• Periféricos: Mouse y Teclado. • Tarjeta de Red. • 521 MB de RAM. • Procesador Pentium 4 (o similar). • 30 GB de espacio en disco.
Origen	Externo al sistema
Artefacto	Todo el sistema
Entorno	Plataforma de recursos Félix Varela.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
Medida de respuesta	
N/A	
Atributo de Calidad	Usabilidad
Sub-atributos/Sub-características	Agradabilidad/ Operabilidad
Objetivo	• El diseño de interfaz debe ser sencillo y fácil de usar, con una iconografía que represente la acción a realizar. Será formal, serio y con una navegación sugerente. • El sistema proporcionará claridad y correcta organización de la información, permitiendo la interpretación correcta e inequívoca de

	ésta. • El sistema podrá ser usado por personas que posean conocimientos básicos de informática y del manejo de ordenadores. • El puntero del mouse cambia y cambia el color del elemento. • Cada campo tiene asociado una pequeña ayuda.
Origen	Interno al sistema/ externo al sistema
Artefacto	Todo el sistema
Entorno	Plataforma de recursos Félix Varela.
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
Medida de respuesta	
N/A	
Atributo de Calidad	Seguridad
Sub-atributos/Sub-características	Acceso restringido
Objetivo	• El sistema debe contar con diferentes niveles de acceso a la información almacenada para garantizar la protección de información de accesos no autorizados. • Mantener el sistema disponible evitando que los mecanismos de seguridad impidan el acceso a la información requerida por los usuarios autorizados. • La comunicación del cliente con el servidor se realiza mediante protocolo HTTPS.
Origen	Interno al sistema/ externo al sistema
Artefacto	Servicios del sistema/ datos del sistema
Entorno	Plataforma de recursos Félix Varela
Estímulo	**Respuesta: Flujo de eventos (Escenarios)**
1.a Acceso no autorizado	
Intentos de acceso a una acción sin privilegios.	No se muestra la lista de recomendaciones si el usuario no está autenticado.
Medida de respuesta	
N/A	

2.7 Historias de Usuario (HU)

Son la técnica utilizada para especificar los requisitos del software. Se trata de una representación en la cual el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (51). A continuación se encuentra un ejemplo de una HU con los parámetros que la definen. Las historias de usuario correspondientes a los restantes requisitos funcionales se muestran en el Anexo #1.

Tabla 4: Descripción de los campos de la HU.

Historia de Usuario	
Número: Número consecutivo a partir del 1.	**Nombre del requisito:** Identifica la HU.
Programador: Persona que implementa el requisito.	**Iteración Asignada:** Precisa la iteración en que será desarrollada la HU.
Prioridad: Define la relevancia e impacto de la HU para el negocio de acuerdo con las necesidades del usuario.	**Tiempo Estimado:** Permite estimar la duración de la implementación, representando con 1, una semana de trabajo.
Riesgo en Desarrollo: Ausencia de miembros del equipo de trabajo y actividades extra productivas y sin carácter investigativo.	**Tiempo Real:** Define el tiempo real de la implementación de la HU.
Descripción: Describe el objetivo, las acciones para lograr el objetivo y el flujo de acciones que se realiza. **1- Objetivo:** Define lo que permite la HU. **2- Acciones para lograr el objetivo (precondiciones y datos):** Define las precondiciones y los datos para que se ejecute el flujo de acciones. **3- Flujo de la acción a realizar:** Describe por orden los pasos a seguir para la ejecución de la HU.	
Observaciones: Se aclaran algunos puntos no mencionados en los campos anteriores sobre la ejecución de la HU.	

Tabla 5: HU_Mostrar recomendaciones basadas en ítem.

Historia de Usuario

Número: 7.	**Nombre del requisito:** Mostrar recomendaciones basadas en ítem.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 2.
Prioridad: Media	**Tiempo Estimado:** 1
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Permite mostrar una lista de recomendaciones basada en los libros visitados por el usuario. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. El usuario debe visitar al menos un libro para recibir las recomendaciones basadas en ítem. **3- Flujo de la acción a realizar:** El usuario accede a la plataforma, una vez en la portada, cumpliéndose las precondiciones anteriores en la parte inferior de la portada se muestra una pestaña con los libros recomendados basados en las visitas del usuario.	
Observaciones: Una vez visitado uno de los libros recomendados este se elimina de la lista de recomendados.	

2.8 Modelo de Diseño

El diseño arquitectónico define la relación entre los elementos estructurales más importantes del software, los estilos arquitectónicos y patrones de diseño que pueden usarse para satisfacer los requisitos definidos para el sistema. El diseño del software es un proceso iterativo mediante el cual los requisitos se traducen en un “plano” para construir el software.

2.8.1 Patrón arquitectónico Modelo-Vista-Controlador (MVC)

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura MVC, que está formado por tres niveles: (52)

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

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación. A continuación se muestra una imagen que representa en funcionamiento del patrón MVC. (52)

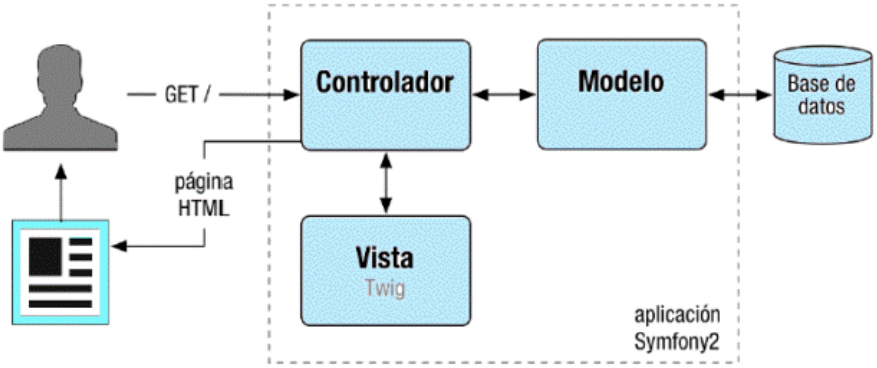


Figura 8: Esquema de funcionamiento de la arquitectura MVC.

2.8.2 Patrones de diseño

Symfony, como framework, hace uso en su implementación de un conjunto de patrones de diseño, los cuales proveen un esquema para refinar los subsistemas y componentes de un sistema de software, o las relaciones entre ellos. “Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software”. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (53). A continuación se describen los patrones utilizados durante el desarrollo del sistema:

Inyección de Dependencias: Es un patrón de diseño orientado a objetos, en el que se suministran objetos a una clase en lugar de ser la propia clase quien cree el objeto. Este patrón es implementado mediante un "contenedor DI"[7]. Dentro del marco de trabajo de Symfony 2 se encuentra la clase Symfony\Bundle\FrameworkBundle\Controller\Controller la cual proporciona un atributo público llamado container, una instancia del contenedor de dependencias. Este permite que desde cualquier

[7] Contenedor de Inyección de dependencias.

clase derivada de la mencionada anteriormente se pueda obtener una instancia del contenedor de dependencias, por lo que se puede instanciar cualquier servicio existente haciendo uso de la función *get* (). A continuación se muestra un ejemplo del uso de este patrón en la investigación:

```
public function getPearsonCorrelation(){
    $userReal = $this->container->get('fortes_user.manager')->getRealUser();

    $id_activo=$userReal->getId();

    $usuarios=$this->container->get('fortes_evaluations.manager')->getUserEvaluator();
    $evaluaciones= $this->container->get('fortes_evaluations.manager')->getAllEvaluations();


    $cant_user= count($usuarios);
    $cant_eval= count($evaluaciones);
    $correlaciones= array();
    $eva_activo= array();
    $user_eval_BD= array();

    for ($i=0; $i< $cant_user; $i++) {
        $userBD=$usuarios[$i];

        for ($j=0; $j< $cant_eval; $j++){

            $var=$evaluaciones[$j];
            $libro= $var->getBook();

            $eva_activo[]= $this->container->get('fortes_evaluations.manager')->getEvaluationE
            $user_eval_BD[]= $this->container->get('fortes_evaluations.manager')->getEvaluatic

        }
        $correlacionAxB= Correlation::pearson($eva_activo, $user_eval_BD);
        if ($usuarios[$i]!=$userReal)
            $correlaciones[]=[$usuarios[$i], $correlacionAxB];

    }
```

Figura 9: Ejemplo del uso del patrón Inyección de Dependencias.

Patrones GRASP (Patrones de asignación de responsabilidades)

Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones; son una serie de buenas prácticas enfocadas a la calidad del software (49). A continuación se describen los patrones GRASP utilizados durante el desarrollo:

Experto: Consiste en que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo o ejecutarlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantiene encapsulada, es decir, disminuye el acoplamiento (54). En el sistema desarrollado se aprecia este patrón en las clases entidades, por ejemplo Book.php es la entidad experta en conocer toda la información referente a los libros.

Controlador: Es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Guía la asignación de responsabilidades relacionadas con la creación de objetos,

permitiendo instanciar y crear las clases que le son necesarias para cumplir sus funcionalidades. La arquitectura MVC brinda una capa específicamente para los controladores, que son el núcleo de este, y especifica la presencia de este patrón (54). En la investigación, el uso de este patrón puede evidenciarse en las clases controladoras ubicadas en *PRFV/BookBundle/Controller*.

Alta Cohesión: Brinda una solución al problema de "asignar una responsabilidad de manera que la cohesión permanezca alta" (49).En el caso del sistema en las clases controladoras se evidencia el uso de este patrón pues las mismas están formadas por varias funcionalidades que están estrechamente relacionadas, siendo las estas las encargadas de definir las acciones y colaborar con otras para realizar tareas que las implican a todas.

Creador: Asigna responsabilidades relacionadas con la creación de objetos o instanciación de nuevos objetos o clases. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento (54). Brinda un soporte a un bajo acoplamiento lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. Este patrón es utilizado en las *clases controladoras* en las cuales se crean objetos de las clases entidades para su manipulación.

Bajo Acoplamiento: Posibilita la idea de tener las clases lo menos relacionadas y en caso de cualquier modificación la repercusión de la misma sea menor potenciando la reutilización (54). Las clases que implementan la lógica de negocio y de acceso a datos se encuentran en el modelo, estas clases no tienen asociaciones con las de la vista o el controlador por lo que la dependencia en este caso es baja, demostrándose así el uso de este patrón. En la solución se puede apreciar que las *clases entidades* son las más reutilizadas.

Patrones GOF

Describen 23 patrones de diseño comúnmente utilizados y de gran aplicabilidad en problemas de diseño usando modelado UML. Se clasifican en tres categorías basadas en su propósito: creacionales, estructurales y de comportamiento (49).

- Los patrones **creacionales** abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados. (Abstract Factory, Factory Method, Prototype y Singleton).

- Los patrones **estructurales** se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades. (Adapter, Decorator, Fachada, y Proxy).
- Los patrones de **comportamiento** están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos. (Chain of Responsability, Interpreter, Observer, Template Method).

A continuación se describen los patrones GOF utilizados durante el desarrollo:

Decorator (Decorador): Es un patrón de tipo estructura, ya que permite determinar que clases y objetos serán utilizados para componer estructuras de mayor tamaño. Este patrón añade dinámicamente nuevas responsabilidades a un objeto. Cada una de las vistas generadas hereda su diseño de la plantilla *"layaut.html.twig"* pues esta almacena el código HTML que es común a todas las páginas de la aplicación, siendo la plantilla contenedora de la estructura y el diseño básico de las vistas.

2.8.3 Diagrama de Clases del Diseño (DCD)

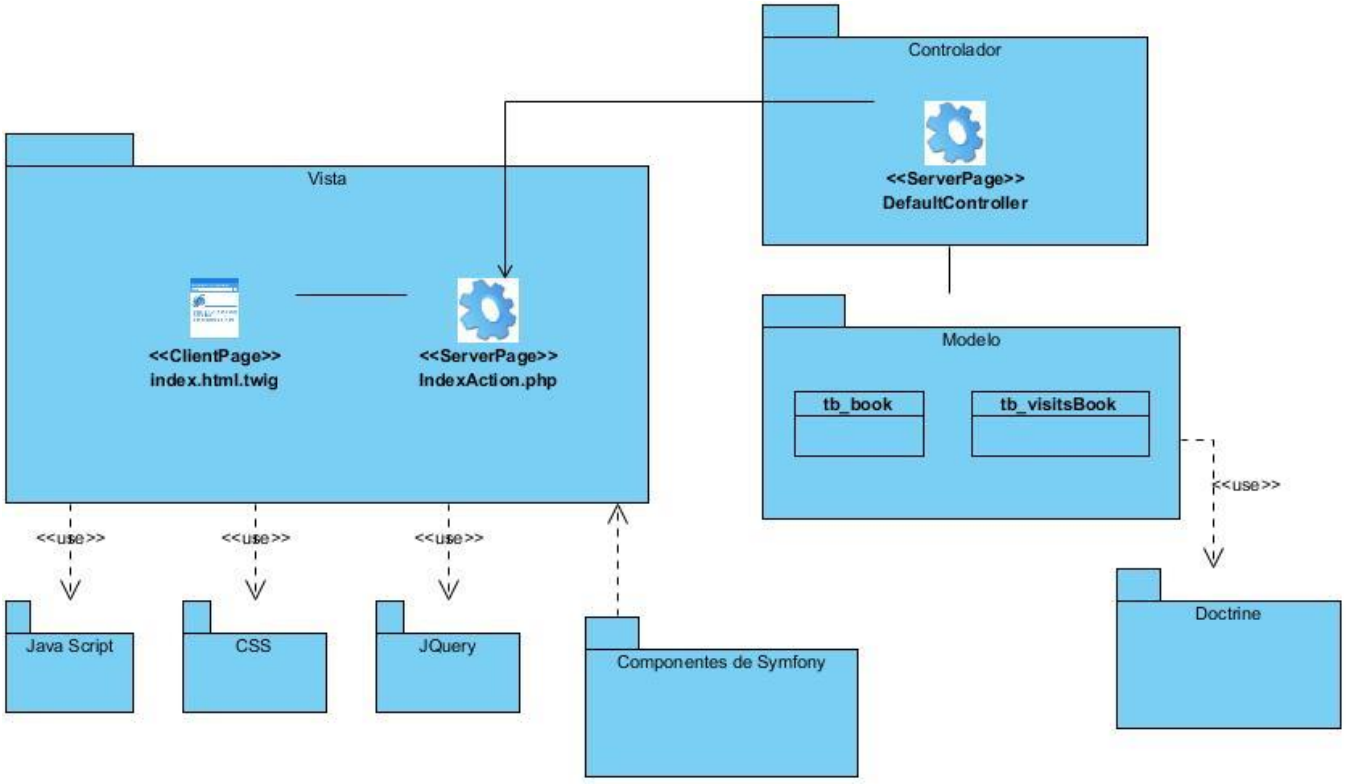


Figura 10: DCD_Mostrar recomendaciones basadas en ítem.

Los restantes diagramas de clases del diseño se muestran en el Anexo #2.

2.8.4 Diagrama de Secuencia (DS)

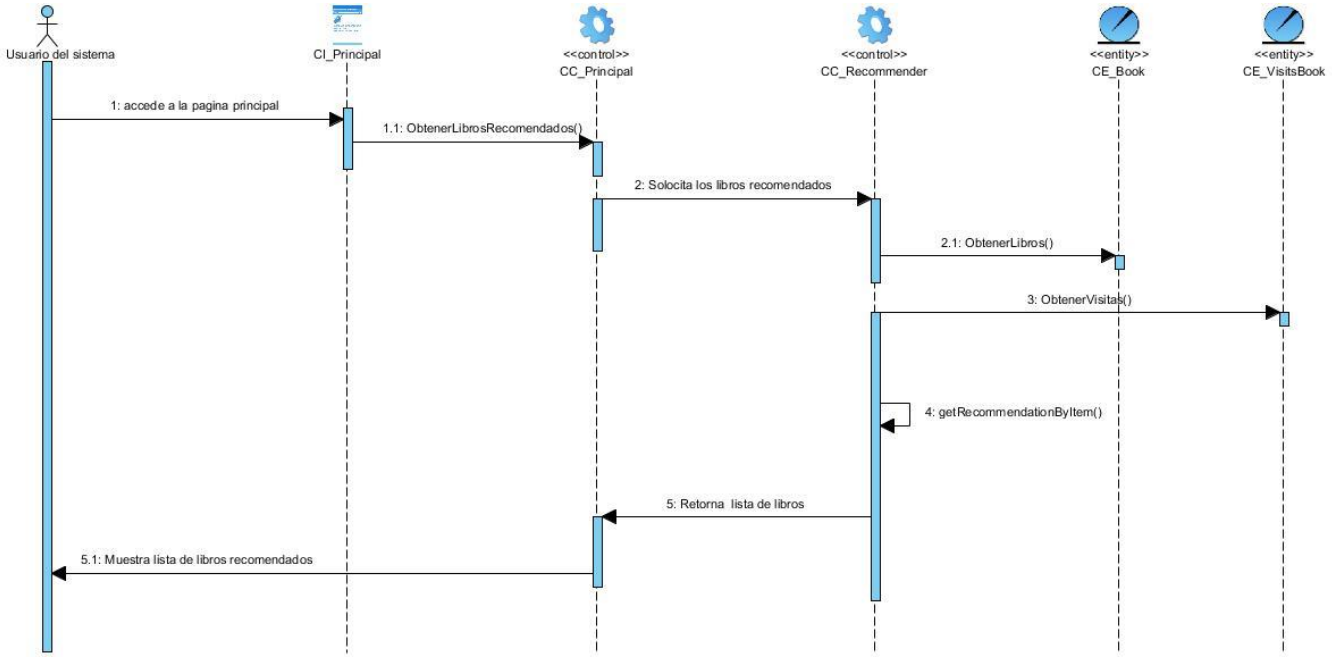


Figura 11: DS_Mostrar recomendaciones basadas en ítem.

Los restantes diagramas de secuencia se muestran en el Anexo #3.

2.8.5 Diagrama de Despliegue

El diagrama de despliegue que se muestra a continuación representa la distribución física del sistema a través de nodos estereotipados. Está compuesto por una PC Cliente que deberá tener instalado un navegador Web, donde la comunicación entre ella y el servidor (Apache) se llevará a cabo a través del protocolo HTTP, en este servidor estará instalada la plataforma y enlazada con su respectiva base de datos.

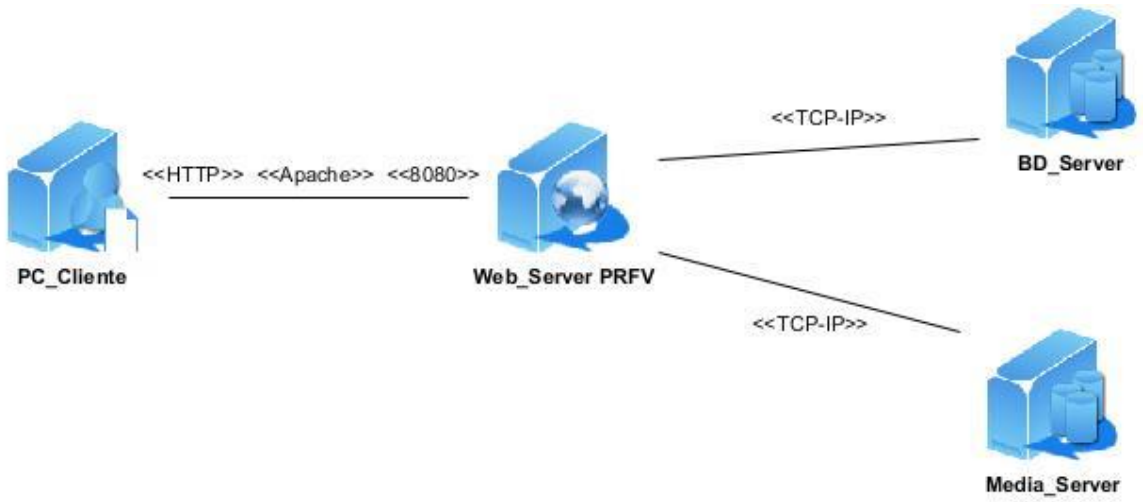


Figura 12: Diagrama de despliegue.

2.8.6 Diseño de la Base de Datos

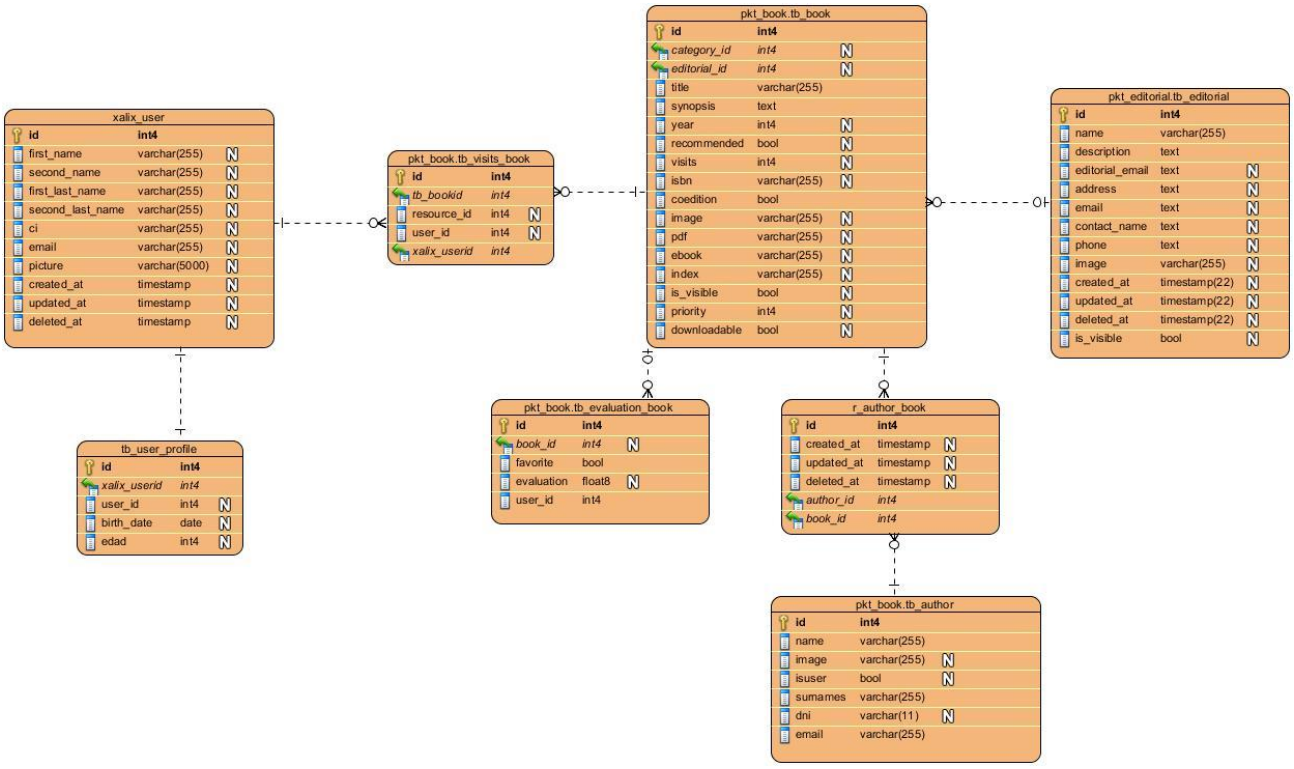


Figura 13: Diagrama Entidad-Relación.

2.8.7 Descripción de las tablas de base de datos

A continuación se muestra la descripción de la tabla **tb_book** creada para almacenar los datos de los libros en el sistema. Las restantes tablas de descripción se muestran en el Anexo #4.

Tabla 6: Descripción de la tabla tb_book de la Base de Datos.

tb_book		
Descripción: Esta tabla es la encargada de almacenar los datos de los libros.		
Atributo	**Tipo**	**Descripción**
Id	Integer(10)	Campo que contiene el identificador del libro.
category_id	Integer(10)	Campo que contiene el identificador de la categoría.
editorial_id	Integer(10)	Campo que contiene el identificador de la editorial
title	Varchar(255)	Campo que contiene el título del libro.
synopsis	Text	Campo que contiene la sinopsis del libro.
year	Integer(10)	Campo que contiene el año de publicación.
recommended	Boolean	Campo que identifica si el libro está recomendado por la editorial o no.
visits	Integer(10)	Campo que contiene la cantidad de visitas a un libro.
isbn	Varchar(255)	Campo que contiene el identificador ISBN del libro.
coedition	Boolean	Campo que identifica si el libro está coeditado o no.
image	Varchar(255)	Campo que contiene la ruta de la foto de portada del libro.
pdf	Varchar(255)	Campo que contiene la ruta del libro en formato pdf.
ebook	Varchar(255)	Campo que contiene la ruta del libro en formato ebooks.
index	Varchar(255)	Campo que contiene la ruta del libro.
is_visible	Boolean	Campo que identifica si el libro está visible o no.
priority	Integer(10)	Campo que contiene la prioridad del libro.
downloadable	boolean	Campo que identifica si el libro se puede descargar o no.

2.9 Conclusiones del capítulo

En este capítulo se ejecutan las tareas requeridas por la fase de Ejecución según la metodología AUP-UCI, haciendo énfasis en los artefactos generados. Se definen los usuarios relacionados con el módulo especificando las funcionalidades de las que dispone. Se definieron y redactaron las historias de usuario, así como las tareas de ingeniería correspondientes a cada uno de los modelos.

Capítulo 3: Implementación y pruebas

3.1 Introducción

Los artefactos generados durante la etapa de análisis y diseño constituyen el paso inicial para el desarrollo de la implementación. El objetivo principal de esta etapa es la generación de clases, componentes u objetos ejecutables que se integrarán a un sistema. Durante el presente capítulo se presenta una descripción detallada a través de los diagramas de componentes del proceso de implementación de las funcionalidades que forman parte de la herramienta a desarrollar. Además se aplicarán las pruebas con el propósito de verificar la calidad del producto.

3.2 Modelo de Implementación

La implementación comienza con el resultado del diseño y se implementa en términos de componentes, es decir, ficheros de código, scripts, ficheros de códigos binarios, ejecutables y similares. Un modelo de implementación incluye suficiente información para construir el sistema. Debe incluir, no solamente la lógica semántica del sistema, los algoritmos, las estructuras de datos y los mecanismos que aseguran el funcionamiento apropiado, sino también las decisiones de organización sobre los artefactos del sistema que son necesarios, permitiendo así el trabajo cooperativo de las personas y el procesamiento por parte de las herramientas (55).

A continuación se describen los artefactos generados durante la implementación definidos por la metodología AUP-UCI en la fase de Cierre.

3.2.1 Diagrama de Componentes

Un diagrama de Componentes ilustra los fragmentos de software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de componentes tiene un nivel de abstracción más elevado que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Los diagramas de componente restantes se muestran en el Anexo #5.

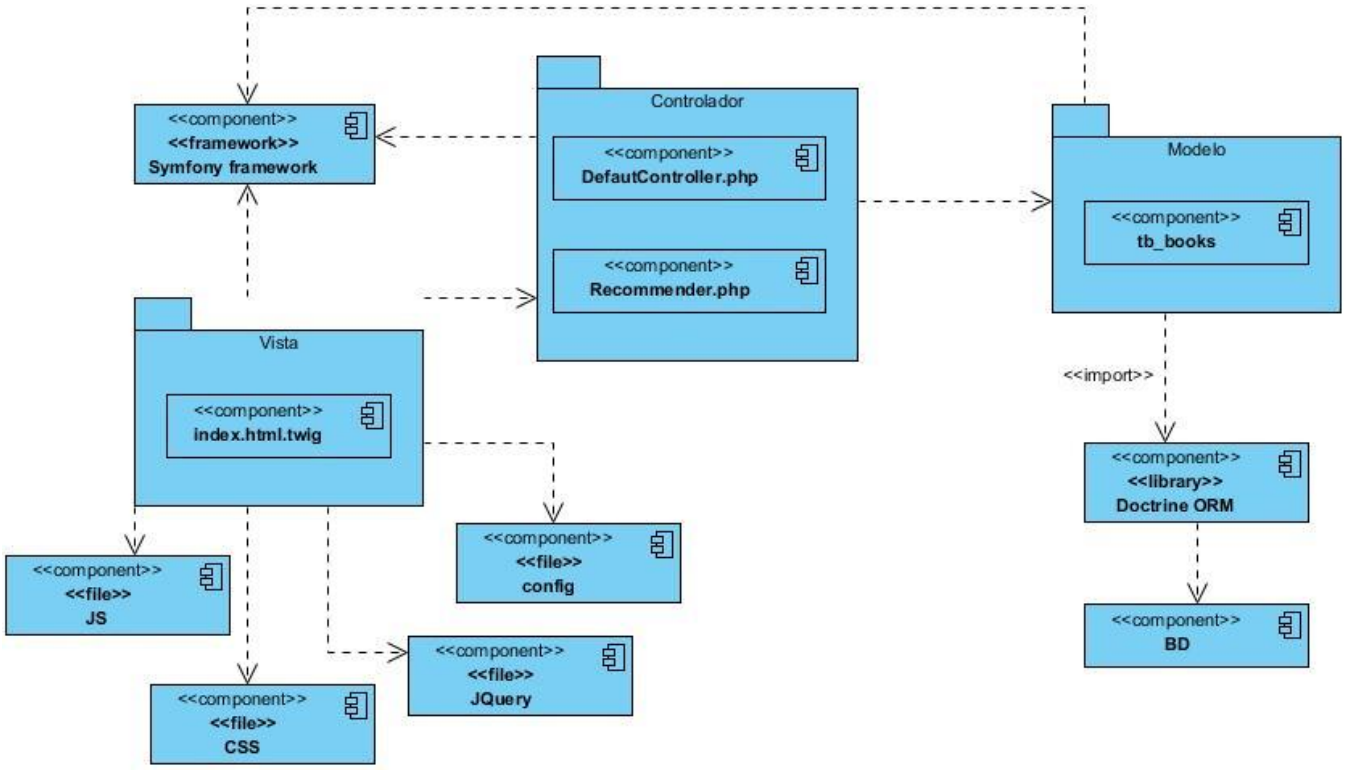


Figura 14: DCOM_Mostrar recomendaciones basadas en ítem.

3.3 Estándares de codificación

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.

La Plataforma de Recursos Félix Varela utilizó en su implementación una serie de estándares de codificación los cuales se van a utilizar también en el desarrollo del módulo recomendador, estos a continuación.

Estructura

- Utilizar las etiquetas cortas (<?).
- Agrega un único espacio después de cada delimitador coma.
- Eliminar espacios después de la apertura de un paréntesis y antes del cierre del mismo.
- Agregar un único espacio alrededor de operadores (==, &&, ...).
- Agregar un único espacio antes de los paréntesis de apertura de una palabra clave de control (if, else, for, while, ...).

- Agregar una línea en blanco antes de la sentencia return.
- Eliminar espacios al final de las líneas.
- Utilizar llaves para indicar el cuerpo de las estructuras de control sin importar el número de sentencias que éstas contengan.
- Colocar las llaves en sus propias líneas para clases, métodos y declaración de funciones.
- Separar las sentencias condicionales y las llaves de apertura con un único espacio sin dejar una línea en blanco.
- Utilizar constantes de tipo PHP nativas en minúsculas: false, true y null. Lo mismo aplica para array().
- Utilizar letras mayúsculas para constantes, con palabras separadas por guiones bajos.
- Definir una clase por archivo.
- Declara las propiedades de las clases antes de los métodos.
- Declarar los métodos públicos primero, luego los protegidos y finalmente los privados.

Convención de nombres

- Utilizar el formato camelCase y no guiones bajos, para variables, funciones y nombres de métodos.
- Utilizar namespace para todas las clases.
- Utilizar Symfony como el namespace de primer nivel para componentes propios de Symfony.
- Utilizar el nombre del Bundle (ejemplo: PRFV/BookRecommenderBundle/) como el namespace de segundo nivel para todos los Bundles.
- Añadir como sufijo Interface a las interfaces.
- Utilizar caracteres alfanuméricos y guiones bajos para nombres de archivos.

Ejemplo de código fuente

En la figura se muestra el código fuente del método getRecommendationByItem(), encargado de recomendar al usuario autenticado los libros similares a los que él ha visitado en algún momento con la característica de que no haya sido visitado por él.

```php
public function getRecomendationByItem(){

    $realUser=$this->container->get('fortes_user.manager')->getRealUser();
    $userId= $realUser->getId();
    $visitsBook= $this->container->get('fortes_book.manager')->getVisitsBooksByUserId($userId);
    $similares= array();

    foreach ($visitsBook as $book)
    {
        $slugBook= $book->getSlug();
        $author=$this->container->get('fortes_author.manager')->getAuthorBySlugBook($slugBook);
        $slugAuthor=$author->getSlug();
        $category= $book->getCategory();
        $editorial= $book->getEditorial();
        $similares= $this->container->get('fortes_book.manager')->getEqualsBooks($slugAuthor, $category, $editorial);
    }

    for ($i=0; $i < count($visitsBook); $i++)
    {
        $libro= $visitsBook[$i];
        for ($j=0; $j < count($similares); $j++)
        {
            $sim= $similares[$j];
            if ($sim->getId() == $libro->getId())
            {
                unset($similares[$j]);
                $similares= array_values($similares);
            }
        }
    }

    return $similares;
}
```

Figura 15: Código de la método getRecommendationByItem().

3.4 Pruebas de software

Las pruebas de software comprenden el conjunto de actividades que se realizan para identificar posibles fallos de funcionamiento, configuración o usabilidad de un programa o aplicación, por medio de pruebas sobre el comportamiento del mismo.

3.4.1 Técnicas de pruebas

Las técnicas más comunes aplicadas a en los procesos de prueba tienen el objetivo de seleccionar buenos casos de prueba, esto es, casos que tengan una probabilidad alta de descubrir un error. Tradicionalmente se han considerado dos enfoques complementarios para seleccionar casos de prueba, denominados caja blanca y caja negra (56). A continuación se explican estas dos técnicas.

Pruebas de Caja Blanca

Las pruebas de caja blanca, en ocasiones llamadas pruebas de caja de vidrio, son una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Las pruebas de caja blanca están dirigidas a las funciones internas del sistema. La prueba es una verificación técnica del software que los desarrolladores

pueden usar para examinar si su código trabaja como se esperaba. Se realizan probando la lógica de la aplicación y comprobando el estado del software en varios puntos, para verificar que los resultados de dicho estado coincidan con los esperados (57).

Pruebas de Caja Negra

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Las pruebas de caja negra no son una alternativa para las técnicas de caja blanca. En vez de ello, es un enfoque complementario que es probable que descubra una clase de errores diferente que los métodos de caja blanca (57).

Dentro de las pruebas de caja negra se incluyen las Técnicas de Prueba que serán descritas a continuación (57):

- **Partición de Equivalencia:** divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.
- **Análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Grafos Causa-Efecto:** permite validar complejos conjuntos de acciones y condiciones.

3.4.2 Diseño de Casos de Prueba (CP)

En los casos de prueba (CP) se incluyen la descripción de los principales escenarios, actores, posibles entradas, variables que intervienen en el proceso y flujo central donde se realiza el procedimiento. A continuación se presenta el CP propuesto para el RF-Generar recomendación basada en ítem. Los casos de pruebas restantes se muestran en el Anexo #6.

Tabla 7: CP RF_Mostrar recomendaciones basadas en ítem.

CP Mostrar recomendaciones basadas en ítem
Descripción: Cuando el usuario accede a la plataforma se le muestra una lista de recomendaciones basada en los libros que ha visitado. **Condiciones de Ejecución:**

El usuario debe estar autenticado.

El usuario debe haber visitado al menos un libro.

SC1: Mostrar recomendaciones basadas en ítem.

Escenario	Descripción	Respuesta del sistema	Flujo básico
EC 1.1: Selecciona la opción **Entrar**.	Una vez en la portada de la aplicación el usuario selecciona la opción **Entrar** que le permite autenticarse en el sistema.	Muestra una página que contiene un formulario para que el usuario se autentique.	Portada/Login
EC 1.2: Inserta los datos y selecciona la opción **Entrar**.	Una vez insertado el usuario y la contraseña selecciona la opción **Entrar.**	Muestra la portada.	Portada/Login/Portada
EC 1.3: Mostrar recomendaciones basadas en ítems.	Una vez que el usuario se autentica se muestra en la portada un carrusel con una lista de recomendaciones	Muestra un carrusel con una lista de recomendaciones basados en los libros visitados por el usuario.	Portada

3.5 Resultados obtenidos por las pruebas

3.5.1 Resultados de las pruebas de Caja Negra

Con el objetivo de verificar el cumplimiento de los requisitos funcionales establecidos para la presente investigación se hace uso de las ***Pruebas de Caja Negra***, teniendo en cuenta la técnica de *partición por equivalencia*. Además, se hace uso de los casos de prueba generados durante este flujo de trabajo con el fin de detectar la mayor cantidad de no conformidades posibles en las funcionalidades del componente realizándose tres iteraciones de prueba. Para el seguimiento de todo el proceso de corrección de no conformidades se realiza una tabla, la misma contará con el Requisito Funcional, la cantidad de no conformidades detectadas (**NC**), la cantidad de no conformidades significativas (**S**) y la cantidad de no significativas (**NS**) por cada Requisito Funcional a que se le realizo el Caso de Prueba:

Tabla 8: Resultados de las pruebas de caja negra por iteraciones.

Requisito Funcional	Iteración 1			Iteración 2			Iteración 3		
	NC	S	NS	NC	S	NS	NC	S	NS
Mostrar recomendaciones basadas en ítem.	3	1	2	2	-	2	1	-	1
Mostrar recomendaciones basadas en el usuario.	4	3	1	3	1	2	2	1	1
Mostrar libros relacionados.	2	-	2	1	-	1	-	-	-
Mostrar libros recomendados por el sistema.	2	-	2	1	1	-	-	-	-
Total	11	4	7	7	2	5	3	1	2

A continuación se muestra un gráfico donde se puntualiza por iteraciones el total de no conformidades identificadas, el total de no conformidades resueltas y la cantidad de no conformidades pendientes.

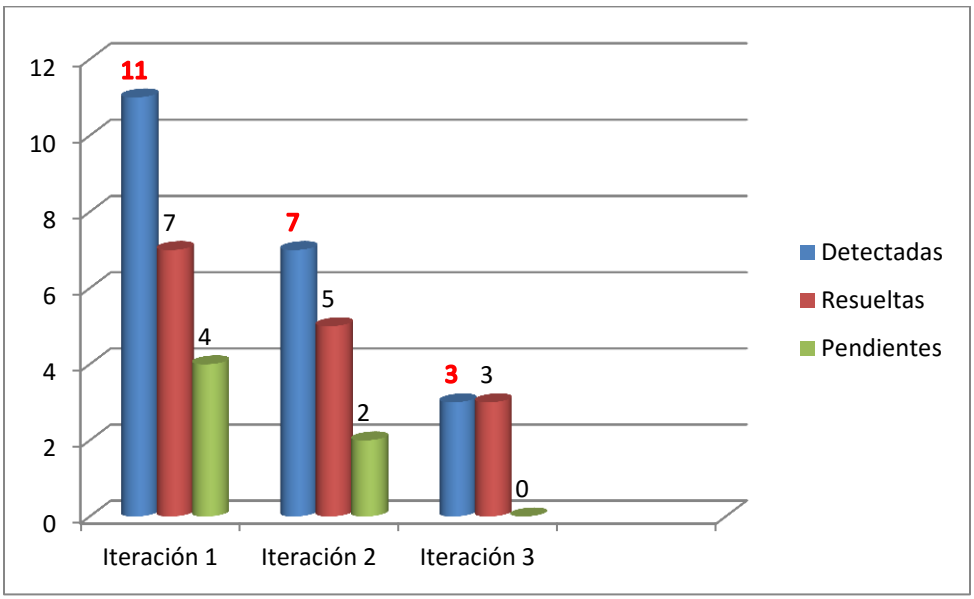


Figura 16: Resultados de las pruebas de caja negra.

3.5.2 Resultados de las pruebas de Caja Blanca

La técnica a aplicar es el camino básico, que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y utilizar esta medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa (57).

Para obtener el conjunto de caminos independientes se construye el grafo de flujo asociado a una función y se calcula su complejidad ciclomática, para este caso se analiza el método *getRecommendationByItem()* encargado de obtener los libros recomendados por el sistema basándose en las visitas del usuario, mostrado en las figuras siguientes. Primeramente se enumeran las sentencias de código.

```
public function getRecomendationByItem(){

    $realUser=$this->container->get('fortes_user.manager')->getRealUser();
    $userId= $realUser->getId();
    $visitsBook= $this->container->get('fortes_book.manager')->getVisitsBooksByUserId($userId);
    $similares= array();

    foreach ($visitsBook as $book)
    {
        $slugBook= $book->getSlug();
        $author=$this->container->get('fortes_author.manager')->getAuthorBySlugBook($slugBook);
        $slugAuthor=$author->getSlug();
        $category= $book->getCategory();
        $editorial= $book->getEditorial();
        $similares= $this->container->get('fortes_book.manager')->getEqualsBooks($slugAuthor, $category, $editorial);
    }

    for ($i=0; $i < count($visitsBook); $i++)
    {
        $libro= $visitsBook[$i];
        for ($j=0; $j < count($similares); $j++)
        {
            $sim= $similares[$j];
            if ($sim->getId() == $libro->getId())
            {
                unset($similares[$j]);
                $similares= array_values($similares);
            }
        }
    }

    return $similares;
}
```

1 2 3 4 5 6 7 8 9 10

Figura 17: Fragmento del código getRecommendationByItem().

Una vez obtenidas las sentencias se construye el grafo, quedando de la forma mostrada en la siguiente figura:

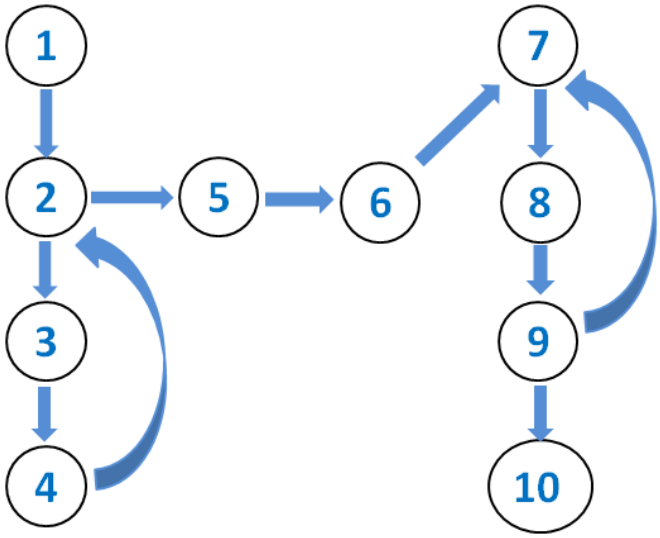


Figura 18: Grafo de flujo asociado a getRecommendationByItem().

La complejidad ciclomática es la métrica de software con que se define la cantidad de caminos independientes de cada una de las funcionalidades del programa y provee el límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez (57).

Se calcula de la siguiente manera:

V(G)=E-N+2 donde E es el número de aristas y N los vértices

V(G)=11-10+2

V(G)=3

Para este caso se obtienen tres posibles caminos independientes y cantidad de pruebas que se deben realizar para comprobar que las sentencias se ejecutan al menos una vez.

Los caminos básicos son:

- Camino básico 1: 1, 2, 3, 4, 2.
- Camino básico 2: 1, 2, 3, 4, 2, 5, 6, 7, 8, 9, 7
- Camino básico 3: 1, 2, 3, 4, 2, 5, 6, 7, 8, 9, 10.

Luego se elaboran los casos de prueba, quedando de la siguiente manera:

- Caso de prueba del camino 1

Condición: Si $visitBook no está vacío.

Resultado esperado: El sistema procede a buscar los libros similares a los visitados por el usuario.

- Caso de prueba del camino 2

Condición: Si el libro visitado se encuentra en la lista.

Resultado esperado: El sistema elimina el libro visitado de la lista de similares, evitando que se recomiende un libro ya visitado.

- Caso de prueba del camino 3

Condición: Si se ejecutan todas las sentencias sin errores.
Resultado esperado: El sistema devuelve una lista de libros similares a los visitados por el usuario.

Se ejecutan los casos de prueba para comparar los resultados obtenidos contra los esperados. Los resultados obtenidos coinciden con los esperados, por lo que se puede asegurar que todas las sentencias del método se han ejecutado al menos una vez.

3.6 Validación experimental de resultados

Para validar los resultados obtenidos por el módulo recomendador desarrollado, el cual permitirá reducir el tiempo de búsqueda de libros según la interacción de los usuarios con la plataforma de recursos Félix Varela; se llevó a cabo la puesta en práctica de un cuasi experimento usando el diseño con posprueba únicamente y grupo de control.

Los **diseños cuasiexperimentales** también manipulan deliberadamente, al menos, una variable independiente para observar su efecto sobre una o más variables dependientes, sólo que difieren de los experimentos “puros” en el grado de seguridad que pueda tenerse sobre la equivalencia inicial de los grupos. En los diseños cuasiexperimentales, los sujetos no se asignan al azar a los grupos ni se emparejan, sino que dichos grupos ya están conformados antes del experimento: son grupos intactos (la razón por la que surgen y la manera como se integraron es independiente o aparte del experimento). Los cuasiexperimentos pueden ser diseñados de diversas formas dependiendo de los elementos con que cuente el o los individuos que lo realicen. (58)

El diseño con posprueba únicamente y grupo de control utiliza dos grupos, donde solo uno recibe el tratamiento experimental. Los grupos son comparados para analizar si el experimento tuvo un efecto sobre la variable dependiente. El diseño y puesta en práctica de este cuasiexperimento se desarrolló a partir de los nueve pasos descritos por Sampieri en su libro Metodología de la investigación. (58)

Fueron definidas una variable independiente y una dependiente, en el caso de la primera refiere al uso del módulo recomendador para mostrar las recomendaciones según las preferencias de los usuarios, en lo adelante X, y la segunda (O) representa el tiempo de búsqueda de los libros según las preferencias del usuario.

La variable independiente tomó dos grados, solo se encuentra en estado de ausencia o presencia. Se usaron tres grupos los cuales se presentan a continuación:

Tabla 9: Descripción de los grupos de estudio.

Grupo	Descripción
G1	Búsqueda de los libros sin el servicio “Búsqueda avanzada”.
G2	Búsqueda de los libros con el servicio “Búsqueda avanzada”.
G3	Búsqueda de libros.

Los grupos G1 y G2 se comportaron como grupos de control y por tanto se sometieron al experimento con ausencia de la variable independiente, mientras que G3 se expuso a su presencia. A continuación, se muestran los valores obtenidos para la variable dependiente en cada uno de los casos a partir de una muestra de 44 libros.

Tabla 10: Resultado del cuasiexperimento entre G1 y G3.

Grupo	Variable independiente(X)	Variable dependiente(O)
G1	__	225 min
G3	X	32 min

Tabla 11: Resultado del cuasiexperimento entre G2 y G3.

Grupo	Variable independiente(X)	Variable dependiente(O)
G2	__	98 min
G3	X	32 min

Una vez analizados los resultados es posible comprobar que con el empleo del módulo recomendador se reduce el tiempo de búsqueda de los libros según las preferencias de los usuarios.

3.7 Conclusiones del capítulo

Como resultado del desarrollo de este capítulo, se generaron los diagramas de componentes, así mismo se realizó la descripción de los casos de prueba con el objetivo de que los mismos sean usados para verificar el correcto funcionamiento de la aplicación. El uso de estándares de codificación aseguró una homogeneidad en la nomenclatura de las clases y métodos facilitando su comprensión a otros programadores. De modo general, se puede afirmar que una vez aplicadas las pruebas los resultados fueron satisfactorios ya que se identificaron a tiempo los problemas presentados por la aplicación y se

les suministró solución de forma inmediata. Finalmente se obtuvo un módulo recomendador de libros para la plataforma de recursos Félix Varela, el cual reducirá el tiempo de búsqueda de libros a los usuarios según su interacción con el sistema.

Conclusiones Generales

Después de desarrollar el presente trabajo y analizar los resultados obtenidos, las conclusiones esenciales a las que se arriban son las siguientes:

- ✓ Los métodos científicos empleados para investigar el objeto de estudio posibilitaron identificar la teoría y los conceptos que sustentan la presente investigación.
- ✓ La selección de AUP-UCI como metodología para guiar el proceso de desarrollo, permitió generar los artefactos fundamentales para lograr un producto de calidad y poder cumplir con las expectativas del cliente.
- ✓ Las herramientas, lenguajes de programación y tecnologías seleccionadas permitieron desarrollar un SR capaz de reducir el tiempo de búsqueda de libros según las preferencias de los usuarios, todo esto haciendo uso de las técnicas y algoritmos como el cálculo de las similitudes a partir de la Correlación de Pearson y el algoritmo KNN.
- ✓ Mediante la realización de las pruebas de Caja Negra y Caja Blanca, se pudo detectar y corregir los errores encontrados, logrando con esto, verificar el perfecto funcionamiento de la solución propuesta.
- ✓ A partir de la puesta en práctica de un cuasiexperimento usando el diseño con posprueba únicamente y grupo de control, quedó validado que el módulo recomendador contribuyó a reducir el tiempo de búsqueda de los libros teniendo en cuenta las preferencias de los usuarios.

Recomendaciones

A partir de la investigación realizada se sugieren las siguientes recomendaciones con el objetivo de que muchas de las consideraciones dadas aquí sean objeto de revisión, de completamiento y de perfeccionamiento en futuros trabajos. Se recomienda:

- Brindar al usuario la opción de conocer cuáles son sus vecinos y a partir de estos mostrar los libros que han visitado.
- Aplicar otras técnicas para calcular la similitud entre usuarios con el fin de obtener valores más exactos, teniendo en cuenta otros criterios de selección.

Referencias bibliográficas

1. **Ostùa, Pablo Navazo.** *El impacto del libro electrónico en el sector literario internacional.* Bologna : s.n., 2011.

2. **Xiao, Bo y Benbasat, Izak.** *E-Commerce Product Recommendation Agent: use, characteristics, and impact.* 2007.

3. **Terveen, Loren y Hill, Will.** *Beyond Recommender Systems: Helping People Help Each Other.* 2001.

4. **Camila Alejandra Cardona Otálora, Saúl Leonardo Celis Valbuena.** *Prototipo de software de un sistema recomendador de productos y servicios en promoción sobre una arquitectura orientada a servicios .* Bogotá, Colombia : s.n., 2015.

5. **Velez-Langs, Oswaldo y Santos, Carlos.** *"Sistemas Recomendadores: Un enfoque desde los algoritmos genéticos".* Lima, Perú : s.n., 2006. Vol. 9.

6. **Adomavicius, Gediminas y Tuzhilin , Alexander.** *Recommendation technologies: Survey of current methods and possible extensions.* University of Minnesota : s.n., 2003.

7. **Gómez, Arianna Mesa y Marcelino Haughton Samada, Ernesto.** *Sistema Recomendador de ejercicios físicos para estudiantes de la Universidad de las Ciencias Informáticas.* La Habana : UCI, 2013.

8. **Paul Resnick, Hal R. Varian.** *Recommender System. Communications of ACM.* 1997.

9. **Peña, Fernando Andres y Riffo Carrillo, Ricardo Elias.** *Revisión, selección e implementacion de un algoritmo de recomendacion de material bibliográfico utlizando tecnología J2EE.* Chile : Universidad del Bio-Bio, 2008.

10. **Cingolani, Lic. Enrique Antonio.** *Evaluación de Sistemas Recomendadores de Contenidos Educativos a través de Estudios de Usuarios.* s.l. : Universidad Abierta Interamericana, 2014.

11. **Escudero, Helena Muñoz.** *Sistema de recomendacion multimedia basado en perfiles de usuarios.* Cataluña : Universdad Politécnica de Cayaluña, 2014.

12. **Iñigo, Luis.** Formación, Empleo y Emprendimiento. [En línea] CEI Formación S.L., 22 de Julio de 2013. [Citado el: 23 de Enero de 2017.] http://ceiformacion.blogspot.com/2013/07/sistemas-recomendadores-iv.html.

13. **Nieves, Ing. Rodrigo Rubén Santiago.** *Modelo de recomendación de problemas a realizar para competir en la Olimpiada de Informática .* Ciudad de México : s.n., 2015.

14. **Capouya, Andres.** Apuntes de Informática. [En línea] 24 de Marzo de 2009. [Citado el: 11 de Noviembre de 2016.] http://informaticafrida.blogspot.com/2009/03/algoritmo.html.

15. **Moya, Ricardo.** Jarroba.com. [En línea] 24 de Septiembre de 2013. [Citado el: 15 de Diciembre de 2016.] https://jarroba.com/sistemas-de-recomendacion-basados-en-filtrado-colaborativo-k-vecinos/.

16. **Julio Vera, Alexander Ocsa Mamani, Kingle Villalba.** *Modelo de sistema de recomendación de Objetos de Aprendizaje en dispositivos móviles, caso: Creación del pensamiento computacional.* 2015.

17. **Oscar Riveros Rey, Juan Romero Fajardo, Jhon Herrera Cubides.** *Implementación de la técnica de los K-Vecinos en un algoritmo recomendador para un sistema de compras utlizando NFC y Android.* s.l. : INGE CUC, 2017.

18. **Nieto, Sergio Manuel Galán.** *Filtrado Colaborativo y Sistemas de Recomendación.* Madrid, España : s.n., 2007.

19. **V.Jensen, Finn y Nielsen, Thomas D.** *Bayesian Networks and Decision Graphs.* s.l. : Springer Science + Business Media, LLC, 2007.

20. **Aciar, Graciela y Aciar, Silvana.** *Recomendador de usuarios en una plataforma colaborativa en base a su perfil y reputación.* San Juan, Argentina : s.n., 2013.

21. **Valdéz, Edward Rolando Núñez.** *Sistemas de Recomendación de contenidos para libros inteligentes.* Oviedo : s.n., 2012.

22. Baquia. [En línea] 3 de Junio de 2016. [Citado el: 20 de Enero de 2017.] https://www.baquia.com/tecnologia/tekstum-un-sistema-de-recomendacion-emocional-de-libros-con-big-data.

23. **Cesé, Yanisel Zamora.** *Sistema recomendación para el catálogo en línea de la bliblioteca de la Universidad de Ciencias Informáticas.* La Habana, Cuba : UCI, 2012.

24. **Carmona, Yaray Hernández y Hernández Orrozco, Yenima.** *Recomendador basado en Filtrado Colaborativo para guiar el aprendizaje interactivo en el módulo Ejercicios de la colección El Navegante.* La Habana, Cuba : UCI, 2013.

25. **Almira, Yanet de los Ángeles Cárdenas.** *Módulo recomendador de libros para la Tienda virtual del Portal CubaLiteraria.* La Habana, Cuba : UCI, 2014.

26. **Sánchez, Tamara Rodríguez.** *Metodología de desarrollo para la Actividad productiva de la UCI .*

27. Mozilla Developer Network. [En línea] [Citado el: 15 de Noviembre de 2016.] https://developer.mozilla.org/es/docs/HTML/HTML5.

28. **González, Enrique.** Aprende a programar. [En línea] [Citado el: 12 de Diciembre de 2016.] http://aprenderaprogramar.com.

29. **Maza, Miguel Ángel Sánchez.** *JavaScript.* 2012.

30. **Huguet, Josep Mir.** *Estudio de los futuros estándares HTML5 y CSS3. Propuesta de actualizacion del sitio www.mpiua.net.* 2012.

31. **Álvarez, Miguel Ángel.** *Manual de JQuery.*

32. **Mestras, Juan Pavón.** *Booststrap 3.0, aplicaciones web.* Madrid : Universidad Complutense Madrid, 2013.

33. **Welling, Like y Thompson, Laura.** *PHP and MySQL Web development.* 2003.

34. *programcion.net.* [En línea] [Citado el: 17 de noviembre de 2016.] http://programacion.net/articulo/10_cosas_sobre_php_7_que_deberias_saber_1086.

35. **Kondas, Arkadiusz.** PHP-ML - Machine Learning library for PHP. [En línea] [Citado el: 22 de Marzo de 2017.] https://php-ml.readthedocs.io.

36. **Iruela, Juan.** Revista Digital Inesem Business School. [En línea] 19 de Enero de 2019. [Citado el: 17 de Noviembre de 2016.] http://revistadigital.inesem.es/nuevas-tecnologias/los-gestores-de-bases-de-datos-mas-usados/.

37. *PostgreSQL.* [En línea] 2 de Octubre de 2010. [Citado el: 17 de Noviembre de 2016.] http://www.postgresql.org.es/sobre_postgresql.

38. **Cruz, Yamilé Peña y Mustelier Ramírez, Franklin.** *Sistema Recomendador de noticias para el portal Octavitos.* La Habana : UCI, 2013.

39. **Raquel Sanchis, Raúl Poler, Ángel Ortiz.** *Técnicas para el modelado de Procesos de Negocio en cadenas de suministro.* Alicante : s.n., 2009.

40. *Fergaciac.* [En línea] 25 de enero de 2013. [Citado el: 22 de noviembre de 2016.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

41. *NetBeans.* [En línea] [Citado el: 22 de Noviembre de 2016.] https://netbeans.org/community/releases/61/index_es.html.

42. *NetBeans.* [En línea] [Citado el: 22 de Noviembre de 2016.] https://netbeans.org/community/releases/80/.

43. *Editores de código.* [En línea] [Citado el: 22 de Noviembre de 2016.] http://www.editoresdecodigo.com/2014/06/descargar-phpstorm-full-ide-para-php-y-mas.html.

44. JetBrains. [En línea] [Citado el: 2016 de Noviembre de 2016.] https://www.jetbrains.com/phpstorm/.

45. *Targetware Informática S.A.C.* [En línea] [Citado el: 22 de noviembre de 2016.] http://www.software.com.ar/p/visual-paradigm-para-uml.

46. **Morell, Yuliexa Batista.** *Componente para la integración de un lector de pantalla con el marco de trabajo Xalix.* La Habana, Cuba : UCI, 2014.

47. **González, Yasmini Cid y Hernández López, Leonel.** *Componente para la gestión de servicios web en el marco de trabajo Xalix.* La Habana, Cuba : UCI, 2014.

48. **Cortez, MCs. Paola Nayeli y De la Cruz Sosa, MCs. Carlos.** Boletín UPIITA. [En línea] [Citado el: 2 de Febrero de 2017.] http://www.boletin.upiita.ipn.mx/index.php/ciencia/510-cyt-numero-39/338-el-perfil-de-usuario.

49. **Larman, Craig.** *UML y Patrones: Introducción al análisis y diseño orientados a objetos.* 1999.

50. **Sosa, Angel Gabriel Olivera y Novelo Can, Carolina.** ScriBD. [En línea] 6 de Septiembre de 2010. [Citado el: 6 de Febrero de 2017.] https://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales.

51. **Letelier Torres, Patricio y Sánchez López, Emilio A.** *Metodologías Ágiles en el desarrollo de Software.* Alicante, España : s.n., 2003.

52. **Potencier, Fabien y Zaninotto, François.** Symfony, la guía definitiva. *LibrosWeb.* [En línea] [Citado el: 5 de Marzo de 2017.] http://librosweb.es/libro/symfony_1_4/capitulo_2/el_patron_mvc.html.

53. **Alberto Cortez, Ana Garis.** *Aplicación de Perfiles UML en la Especificación de Patrones de Comportamiento.* Argentina : s.n., 2012.

54. **Carmona, Juan García.** *Solid y GRASP. Buenas prácticas hacia el éxito en el desarrollo de software .* 2012.

55. **James Rumbaugh, Ivar Jacobson, Grady Booch.** *El lenguaje unificado de modelado. Manual de referencia.* s.l. : Addison Wesley, 1998.

56. **Javier Tuya, Isabel Ramos Román, Jose Javier Dolado Cosín.** *Técnicas cuantitativas para la gestión en la ingeniería de software.* 2007.

57. **Pressman, Roger S.** *Ingeniería del software. Un enfoque practico.* 2000.

58. **Sampieri, Roberto Hernádez.** *Metodología de la investigacion.* Mexico : s.n., 6ta edición.

59. **López, Lic. Rodney Rodríguez y Dr. Juilo Vidal, Larramendi.** *Sistema recomendador y adaptativo para bibliotecas virtuales.* La Habana : Universidad de la Habana, Cuba.

60. **Otálora, Camila Alejandra Cardona y Celis Valbuena, Seul Leonardo.** *Prototipo de software de un sistema recomendador de productos y servicios en promoción sobre una arquitectura orientada a servicios.* Bogotá, Colombia : s.n., 2015.

Anexos

Anexo #1: Historias de Usuario.

Historia de Usuario	
Número: 1.	**Nombre del requisito:** Crear perfil de usuario.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 1.
Prioridad: Alta.	**Tiempo Estimado:** 1.
Riesgo en Desarrollo: N/A.	**Tiempo Real:** N/A.
Descripción: **1- Objetivo:** Permite crear automáticamente el perfil de usuario según su navegación por el sistema. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. **3- Flujo de la acción a realizar:** Una vez autenticado el usuario se crea el perfil automáticamente y se va almacenando sus acciones en la Base de Datos para su posterior uso.	
Observaciones: La actividad del usuario se almacena en las tablas tb_visits y tb_evaluations.	

Historia de Usuario	
Número: 2	**Nombre del requisito:** Calcular índice de correlación de Pearson.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 1.
Prioridad: Alta.	**Tiempo Estimado:** 2.
Riesgo en Desarrollo: N/A.	**Tiempo Real:** N/A.HU.
Descripción: **1- Objetivo:** Calcula el índice de correlación de Pearson del usuario activo con los usuarios que han evaluado al menos un libro. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado.	

El usuario autenticado debe haber evaluado al menos un libro.

3- Flujo de la acción a realizar:

Una vez autenticado el usuario el sistema calcula su índice de correlación a partir de las evaluaciones a los libros que ha visitado. El sistema devuelve una lista con los usuarios correlacionados y su respectivo índice de correlación.

Observaciones

Historia de Usuario	
Número: 3.	**Nombre del requisito:** Establecer usuarios semejantes.
Programador: Yuniel Lavin Gé.	**Iteración Asignada:** 1.
Prioridad: Alta.	**Tiempo Estimado:** 1.
Riesgo en Desarrollo: N/A.	**Tiempo Real:** N/A.

Descripción:

1- Objetivo:

Obtener una lista de los usuarios más semejantes a usuario autenticado.

2- Acciones para lograr el objetivo (precondiciones y datos):

El usuario debe estar autenticado.

3- Flujo de la acción a realizar:

Una vez ejecutada la segunda Historia de Usuario la cual devuelve una lista de usuarios correlacionados, se ordena esta de mayor a menor según el índice de correlación y obteniéndose un máximo de 10 usuarios semejantes.

Observaciones: Una vez ordenada la lista se seleccionan los 10 usuarios con el índice de correlación más alto.

Historia de Usuario	
Número: 4.	**Nombre del requisito:** Establecer ítems semejantes.
Programador: Yuniel Lavin Gé.	**Iteración Asignada:** 1.
Prioridad: Alta.	**Tiempo Estimado:** 1.
Riesgo en Desarrollo N/A.	**Tiempo Real:** N/A.

Descripción:

1- Objetivo: Devuelve una lista de libros semejantes según su categoría, autor y editorial. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe visitar al menos un libro. **3- Flujo de la acción a realizar:** El usuario visita un libro y el sistema analiza los datos del libro y devuelve una lista de libros con igual categoría, autor y editorial respecto al libro visitado.
Observaciones:

Historia de Usuario	
Número: 5.	**Nombre del requisito:** Calcular predicción de evaluación por usuario.
Programador: Yuniel Lavin Gé.	**Iteración Asignada:** 2.
Prioridad: Alta.	**Tiempo Estimado:** 2.
Riesgo en Desarrollo: N/A.	**Tiempo Real** N/A.
Descripción: **1- Objetivo:** Calcular la predicción de evaluación que le daría el usuario autenticado a un libro que no ha evaluado. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. **3- Flujo de la acción a realizar:** Una vez autenticado el usuario, el sistema obtiene la lista de usuarios semejantes al autenticado y una lista de libros no visitados. Posteriormente a partir de la fórmula de Media Ponderada se calcula la evaluación que le daría el usuario autenticado a un libro que no ha evaluado. El sistema devuelve una lista de libros con su respectiva predicción.	
Observaciones:	

Historia de Usuario	
Número: 7.	**Nombre del requisito:** Generar recomendaciones basadas en ítem.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 2.
Prioridad: Alta	**Tiempo Estimado:** 1

Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Devuelve una lista de recomendaciones basada en los libros visitados por el usuario. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. El usuario debe visitar al menos un libro para generar las recomendaciones basadas en las visitas a los libros. **3- Flujo de la acción a realizar:** El usuario al visitar un libro este se almacena en la Base de Datos, posteriormente el sistema determina los libros similares a los visitados por el usuario y devuelve una lista con un máximo de 10 libros.	
Observaciones:	

Historia de Usuario	
Número: 8.	**Nombre del requisito:** Generar recomendaciones basadas en el usuario.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 2.
Prioridad: Alta	**Tiempo Estimado:** 1
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Devuelve una lista de recomendaciones basada en las predicciones de evaluación del usuario autenticado. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. **3- Flujo de la acción a realizar:** El sistema obtiene una lista de predicciones de evaluación a partir de la quinta Historia de Usuario, posteriormente la ordena teniendo en cuenta la evaluación y devuelve la lista con un máximo de 10 libros.	
Observaciones:	

Historia de Usuario	
Número: 9.	**Nombre del requisito:** Mostrar recomendaciones basadas en el usuario.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 3.

Prioridad: Media.	**Tiempo Estimado:** 1
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Se muestra una lista de recomendaciones basadas en los libros visitados por el usuario.

2- Acciones para lograr el objetivo (precondiciones y datos):

El usuario debe estar autenticado.

3- Flujo de la acción a realizar:

En la portada se muestra una lista de recomendaciones basadas en las visitas del usuario, para mostrar la recomendación se apoya en la HU_03, HU_05 y la HU_08. Se muestra un máximo de 10 recomendaciones.

Observaciones:

Historia de Usuario	
Número: 10.	**Nombre del requisito:** Mostrar libros relacionados.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 3.
Prioridad: Media	**Tiempo Estimado:** 1.
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A

Descripción:

1- Objetivo:

Muestra una lista de libros relacionados al libro visitado.

2- Acciones para lograr el objetivo (precondiciones y datos):

El usuario debe estar autenticado.

3- Flujo de la acción a realizar:

Flujo 1: En la portada se muestra una lista de recomendaciones, cuando se selecciona la opción **Ver** la portada del libro se muestra una ventana con los principales datos del libro y en la parte inferior se muestra una lista de libros relacionados, es decir libros similares, esta Historia de Usuario se apoya en la HU_04.

Flujo 2: Al visitar un libro en la parte inferior de la ventana se muestran tres pestañas, **Preguntas**, **Índice** y **Libros relacionados**, en esta última se muestra una lista de libros semejantes al libro visitado.

Historia de Usuario	
Número: 11.	**Nombre del requisito:** Mostrar libros recomendados por el sistema.
Programador: Yuniel Lavin Gé	**Iteración Asignada:** 3.
Prioridad: Alta	**Tiempo Estimado:** 1.
Riesgo en Desarrollo: N/A	**Tiempo Real:** N/A
Descripción: **1- Objetivo:** Muestra una lista de recomendaciones basada en los libros visitados por el usuario y en los usuarios semejantes al usuario autenticado. **2- Acciones para lograr el objetivo (precondiciones y datos):** El usuario debe estar autenticado. **3- Flujo de la acción a realizar:** En la portada se muestra una lista de libros recomendados apoyado en las HU_06 y HU_08	
Observaciones: Las recomendaciones no exceden de 20 libros.	

Anexo #2: Diagramas de Clases del Diseño (DCD)

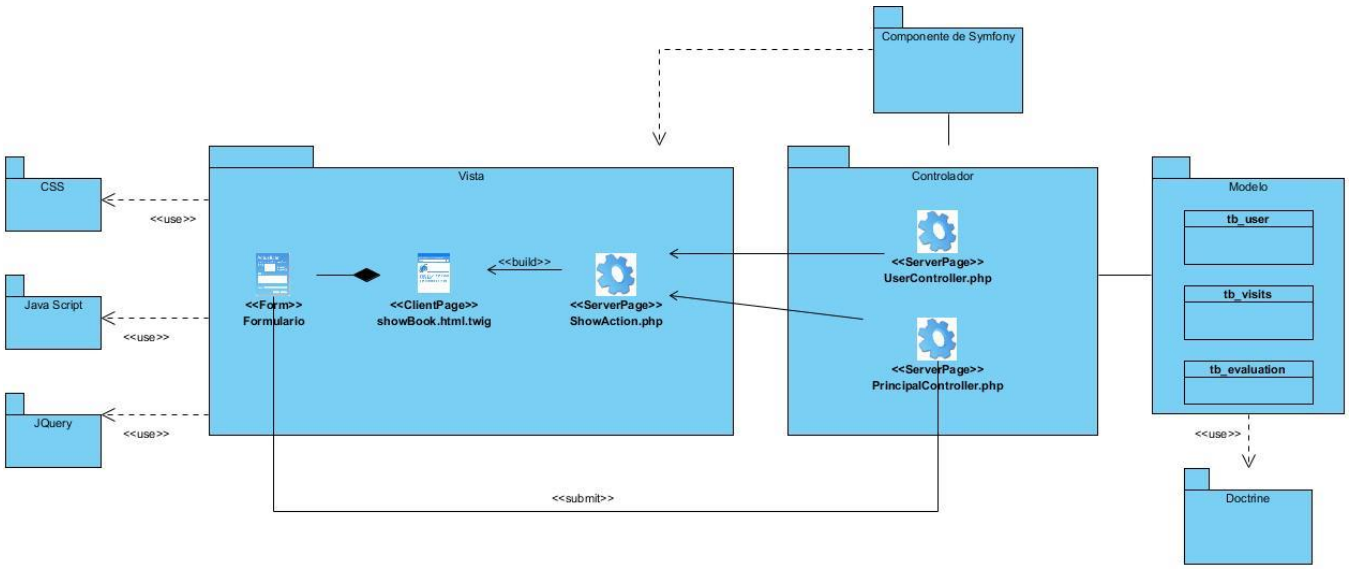


Figura 19: DCD Crear perfil de usuario.

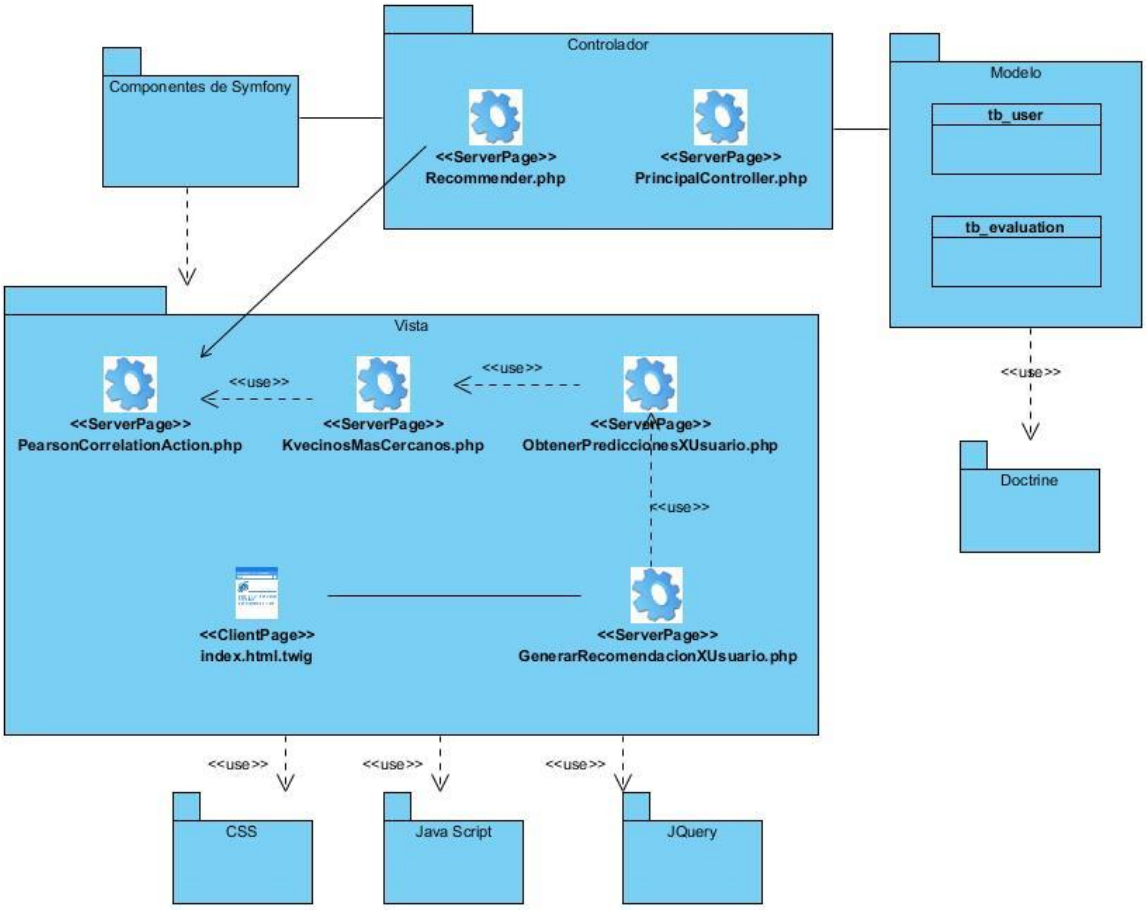


Figura 20: DCD Generar recomendaciones basadas en el usuario.

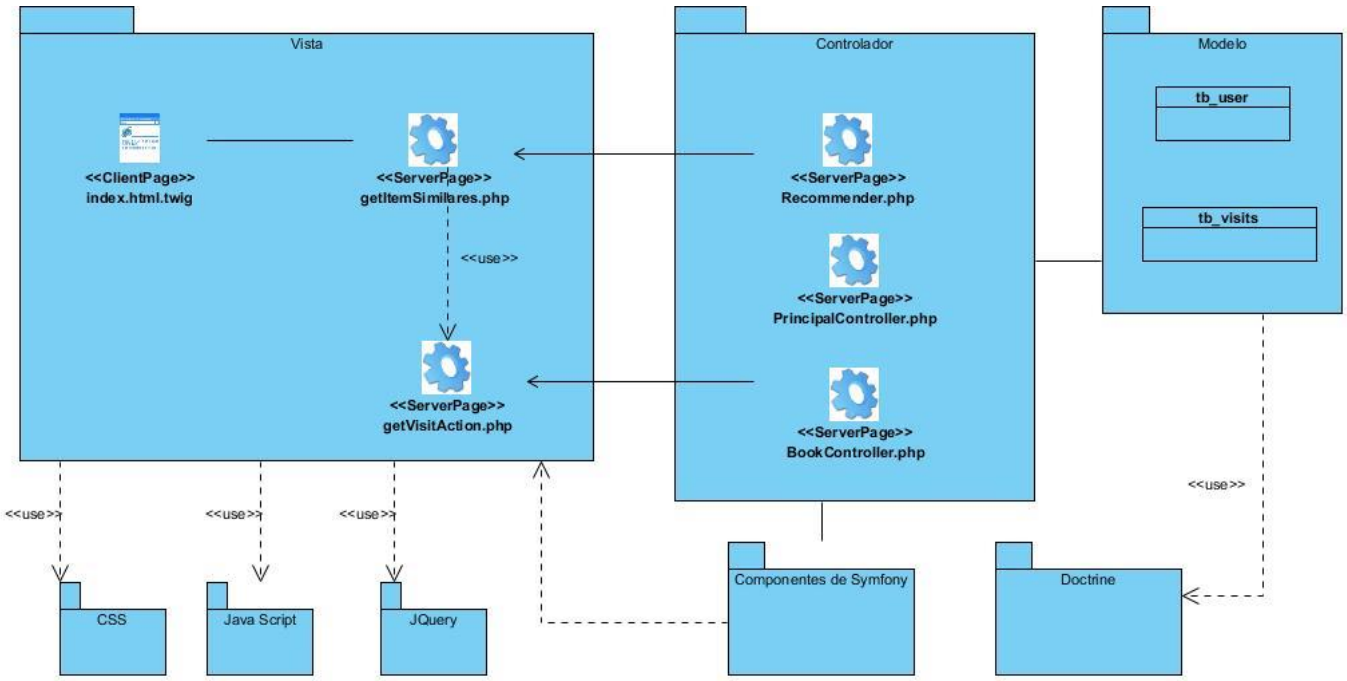


Figura 21: DCD Generar recomendaciones basadas en ítem.

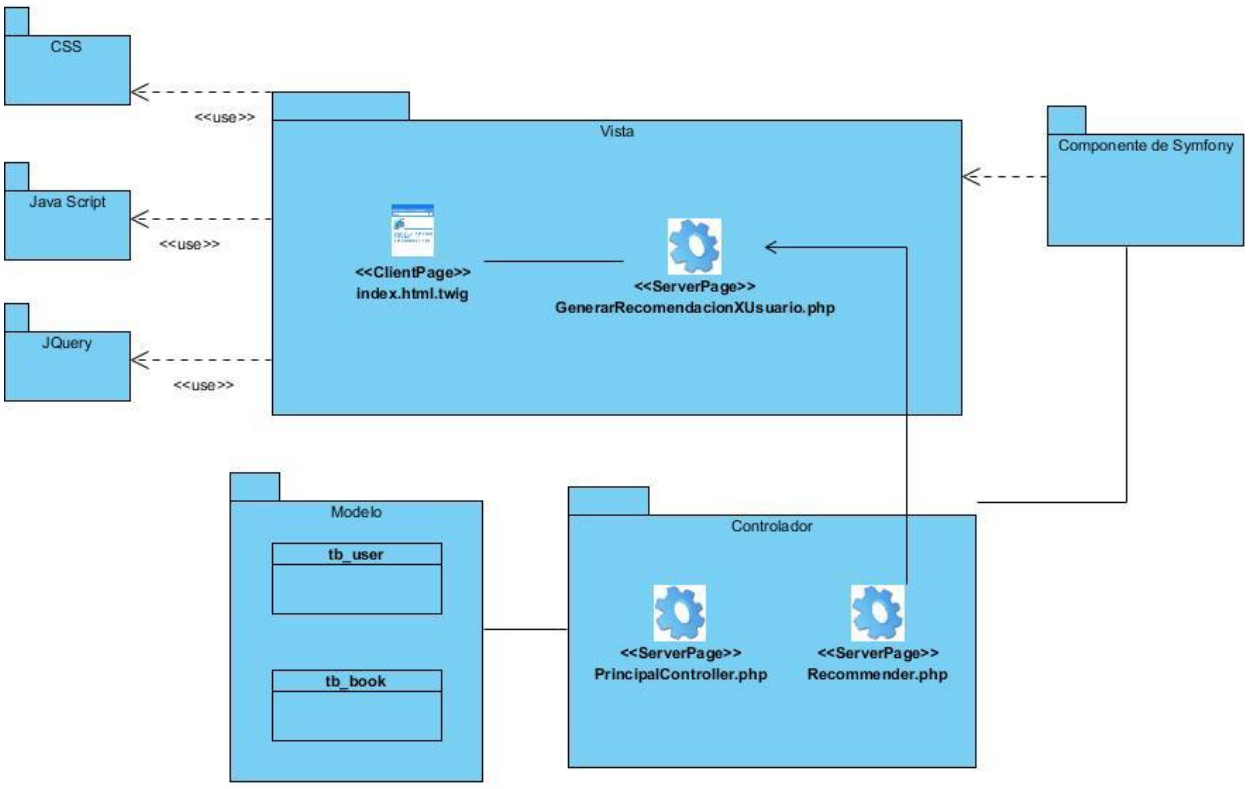


Figura 22: DCD Mostrar recomendaciones basadas en el usuario.

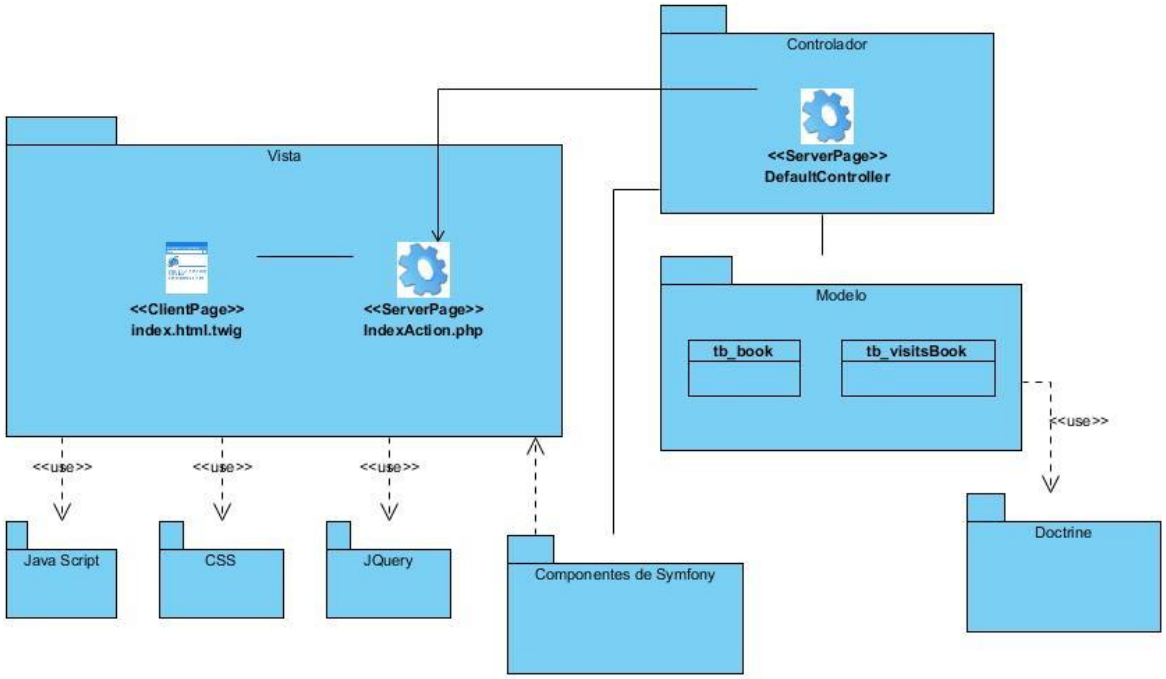


Figura 23: DCD Mostrar recomendaciones basadas en ítem.

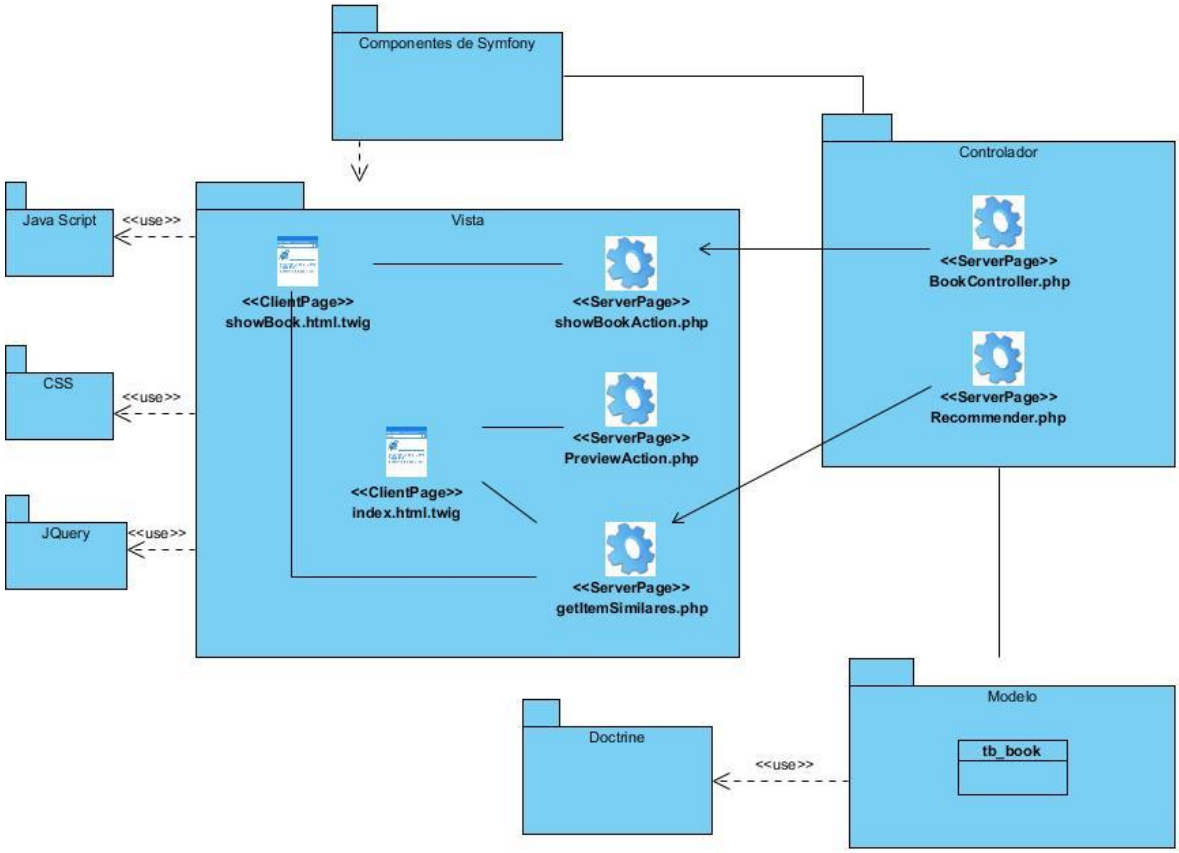


Figura 24: DCD Mostrar libros relacionados.

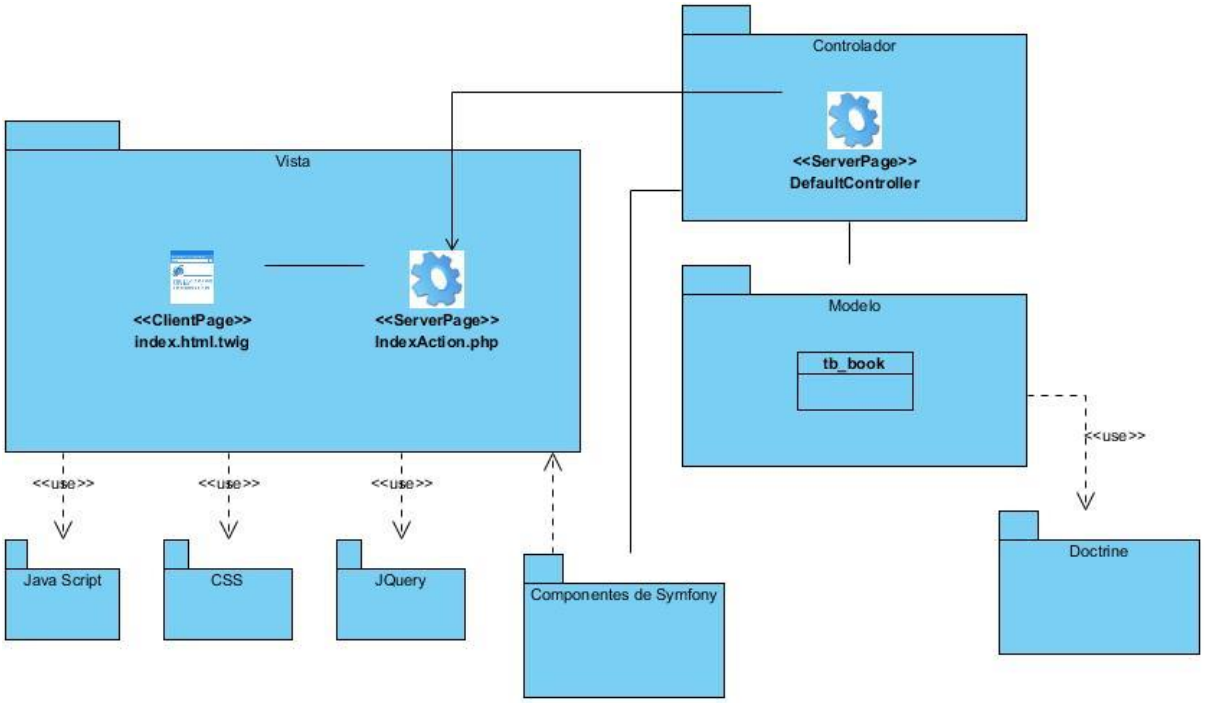


Figura 25: DCD Mostrar libros recomendados por el sistema.

Anexo #3: Diagramas de secuencia (DS)

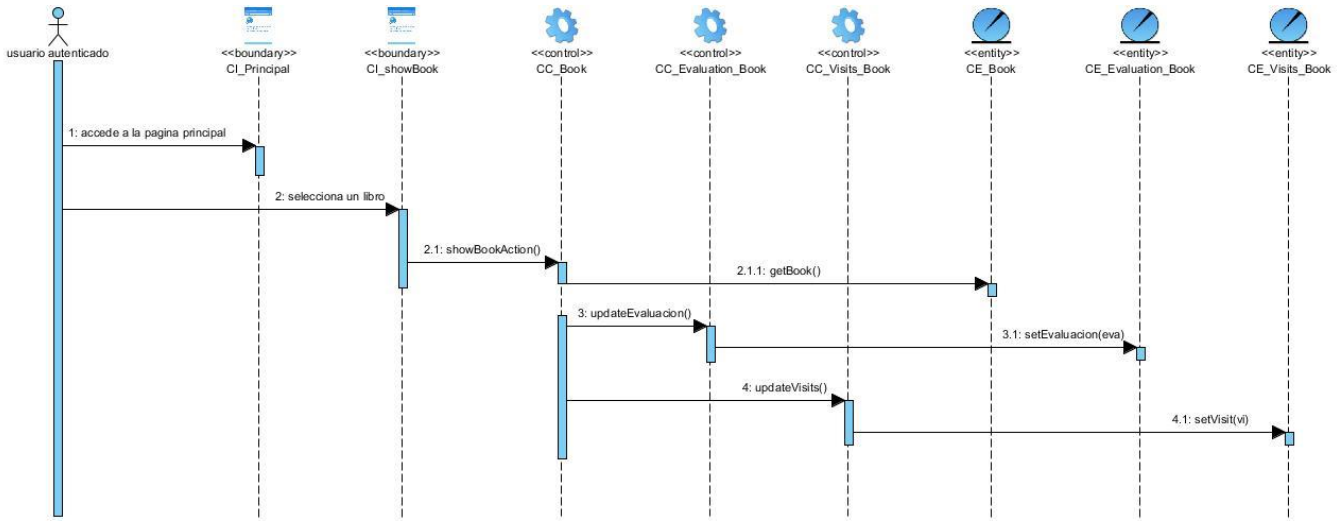


Figura 26: DS Crear perfil de usuario.

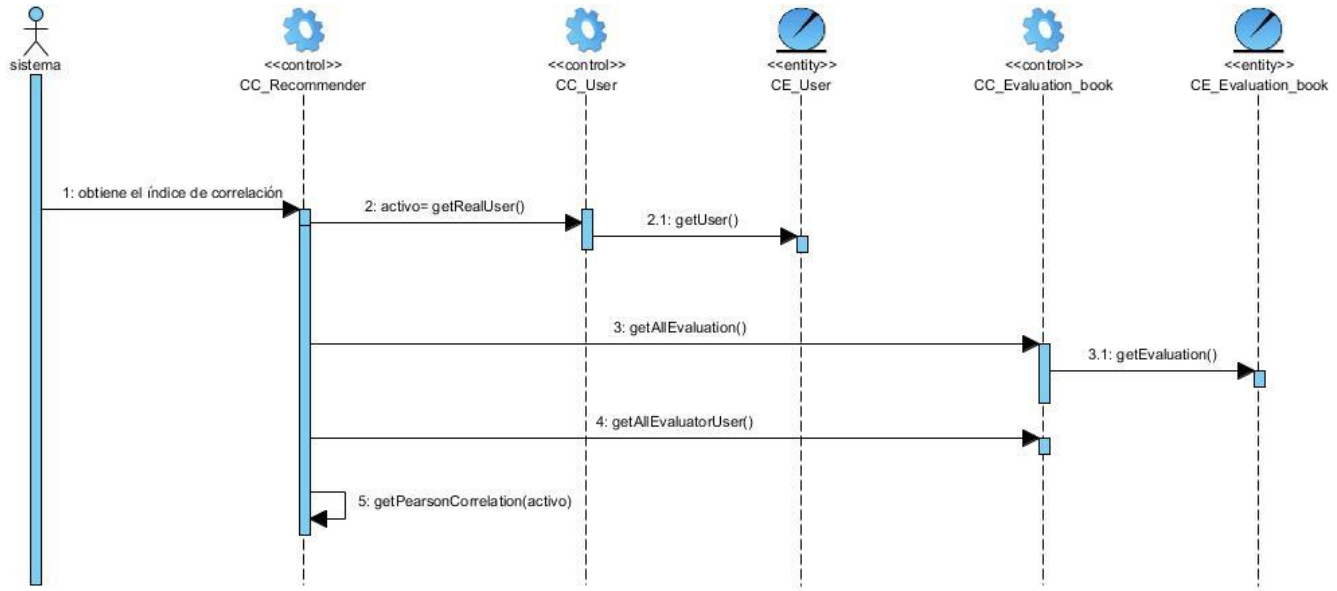


Figura 27: DS Calcular índice de correlación de Pearson.

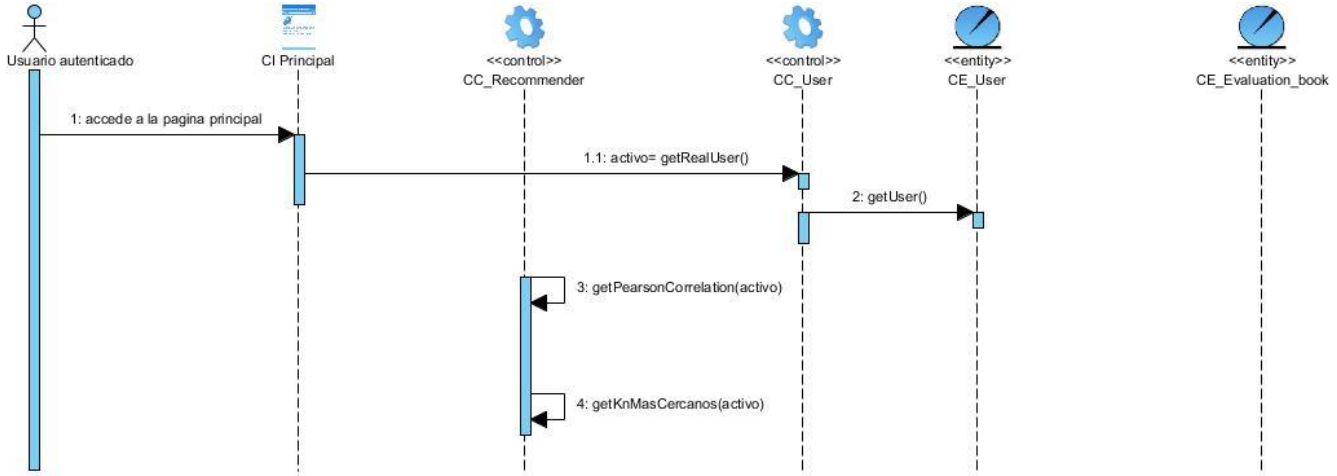


Figura 28: DS Establecer usuarios semejantes.

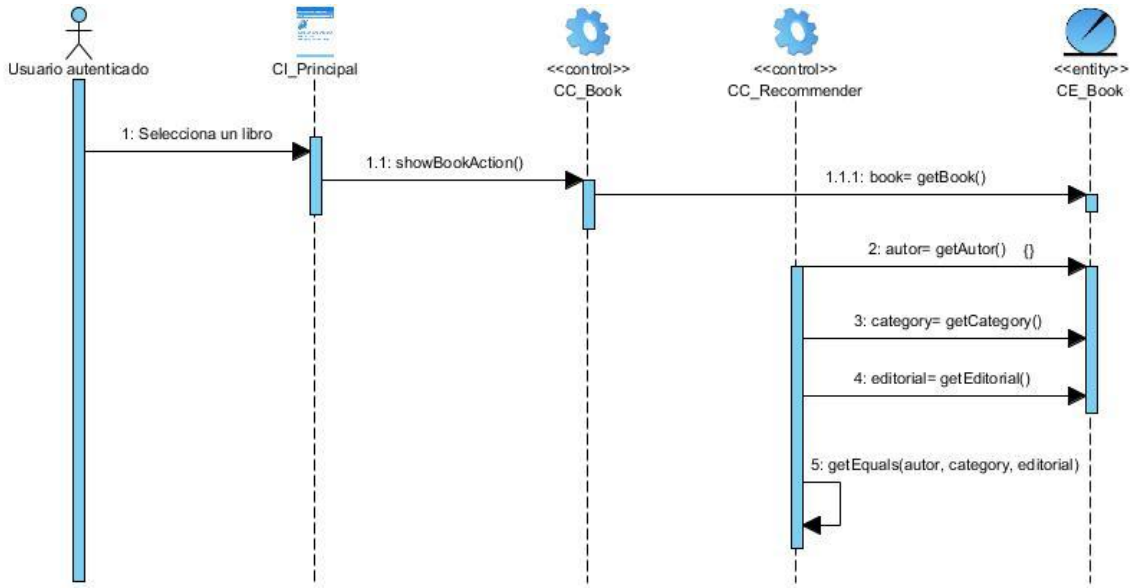


Figura 29: DS Establecer ítems semejantes.

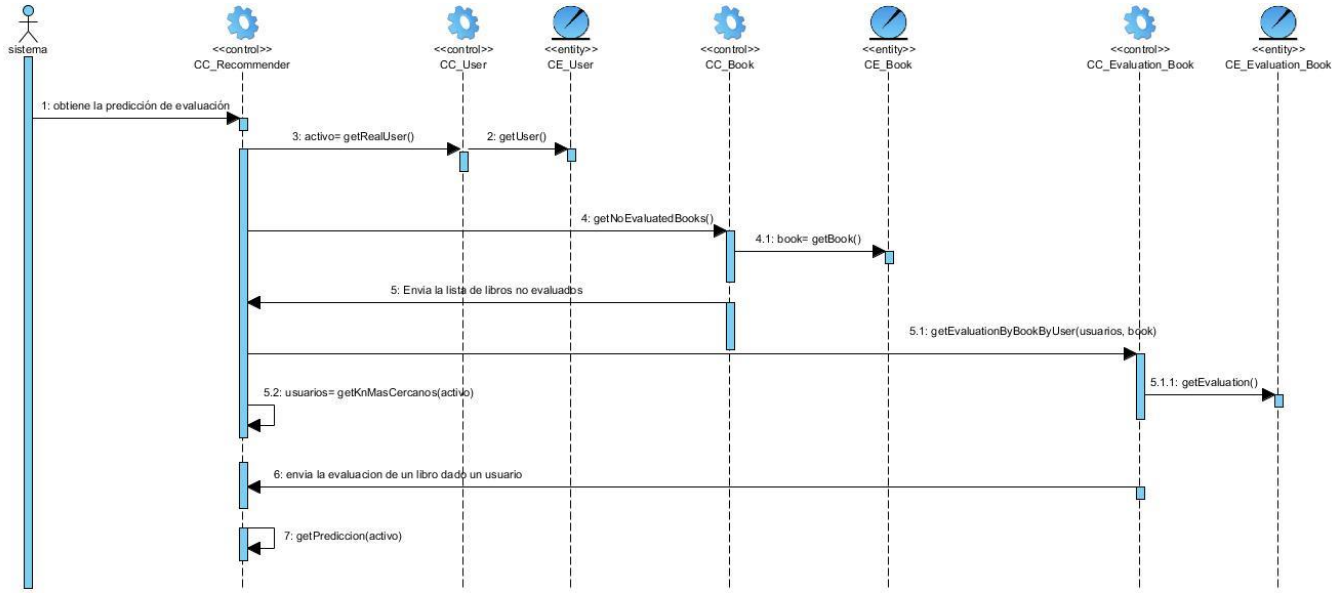


Figura 30: DS Calcular predicción de evaluación por usuario.

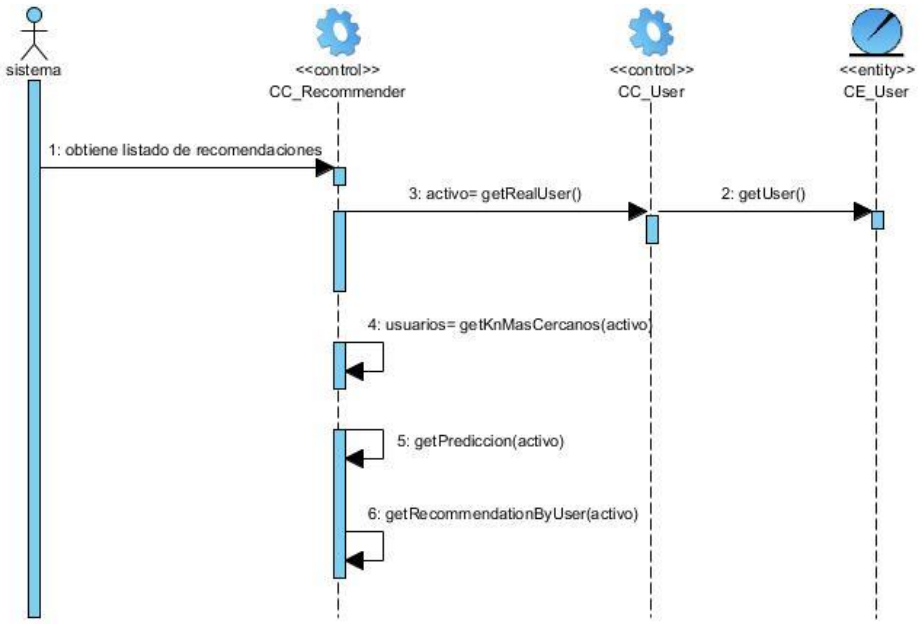


Figura 31: DS Generar recomendaciones basadas en el usuario.

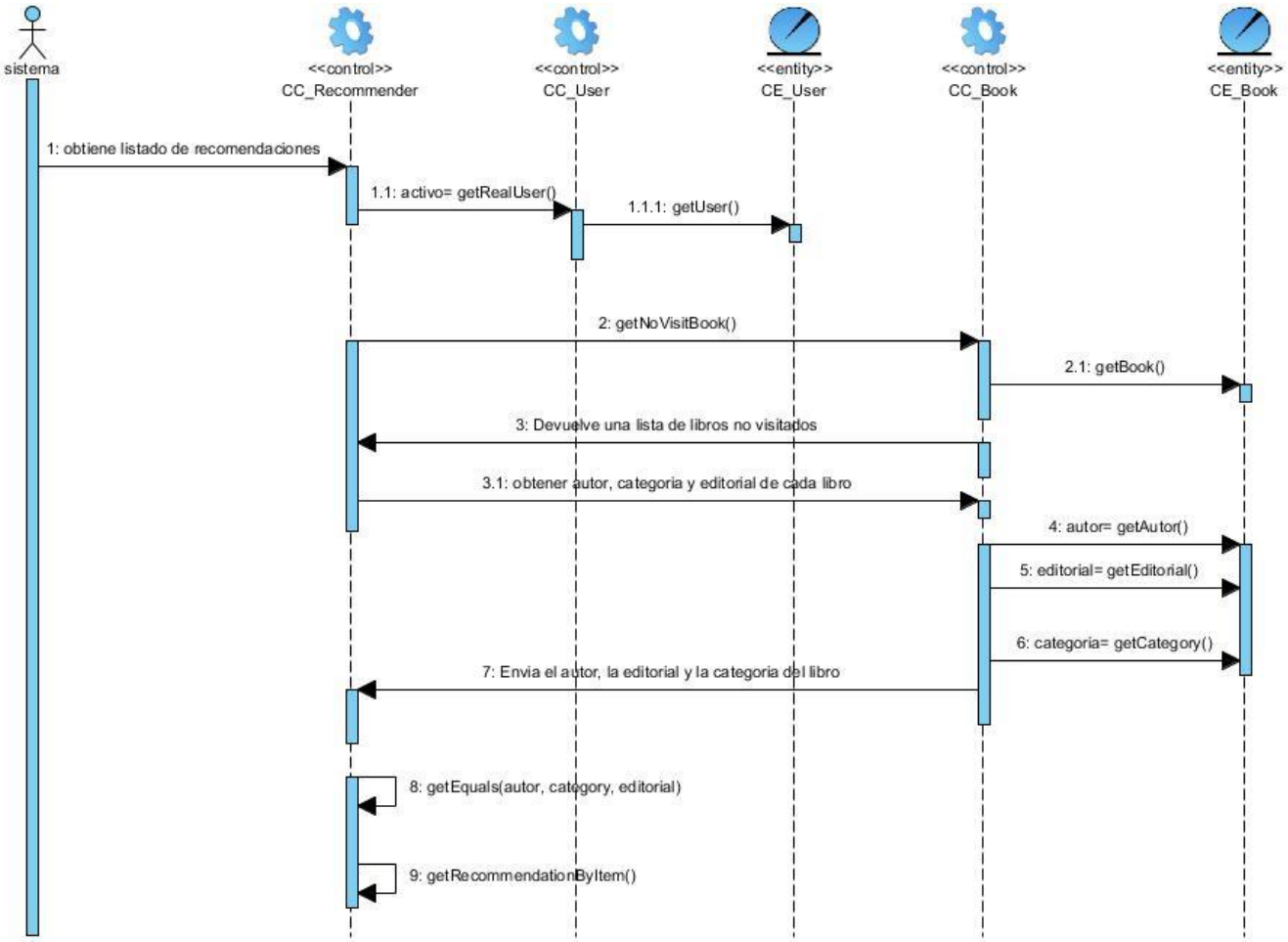


Figura 32: DS Generar recomendaciones basadas en ítem.

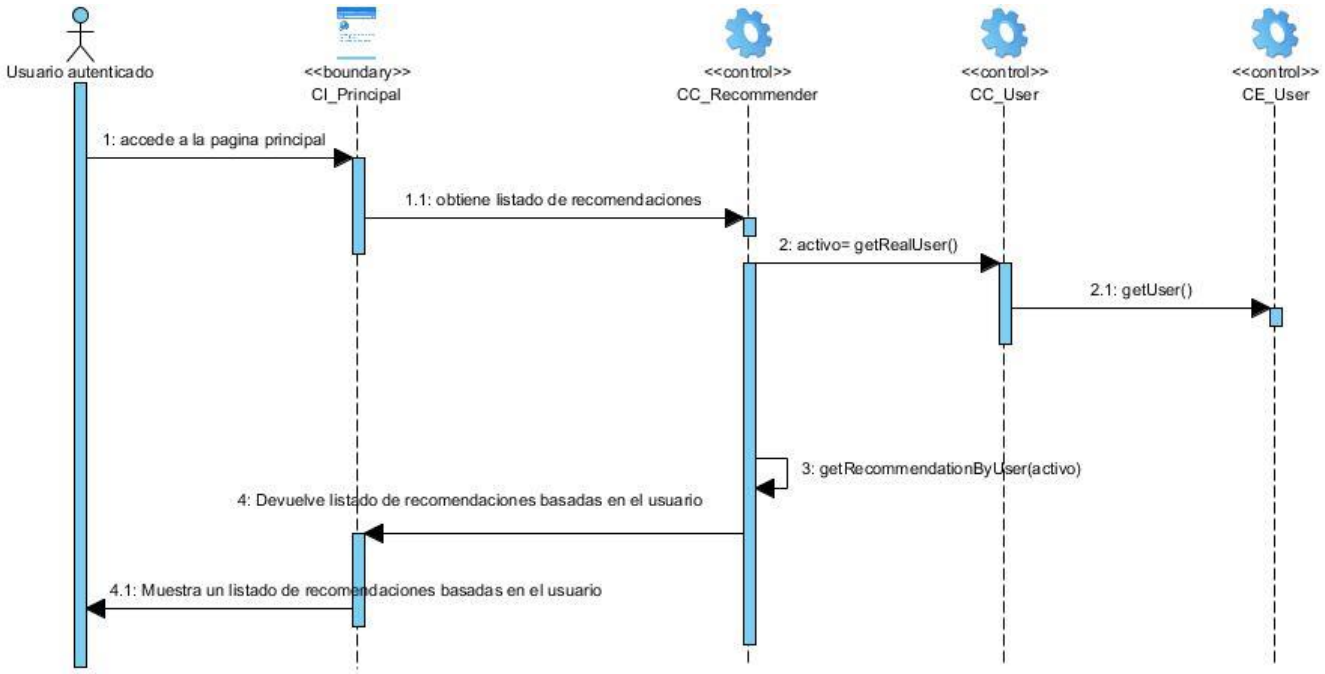


Figura 33: DS Mostrar recomendaciones basadas en el usuario.

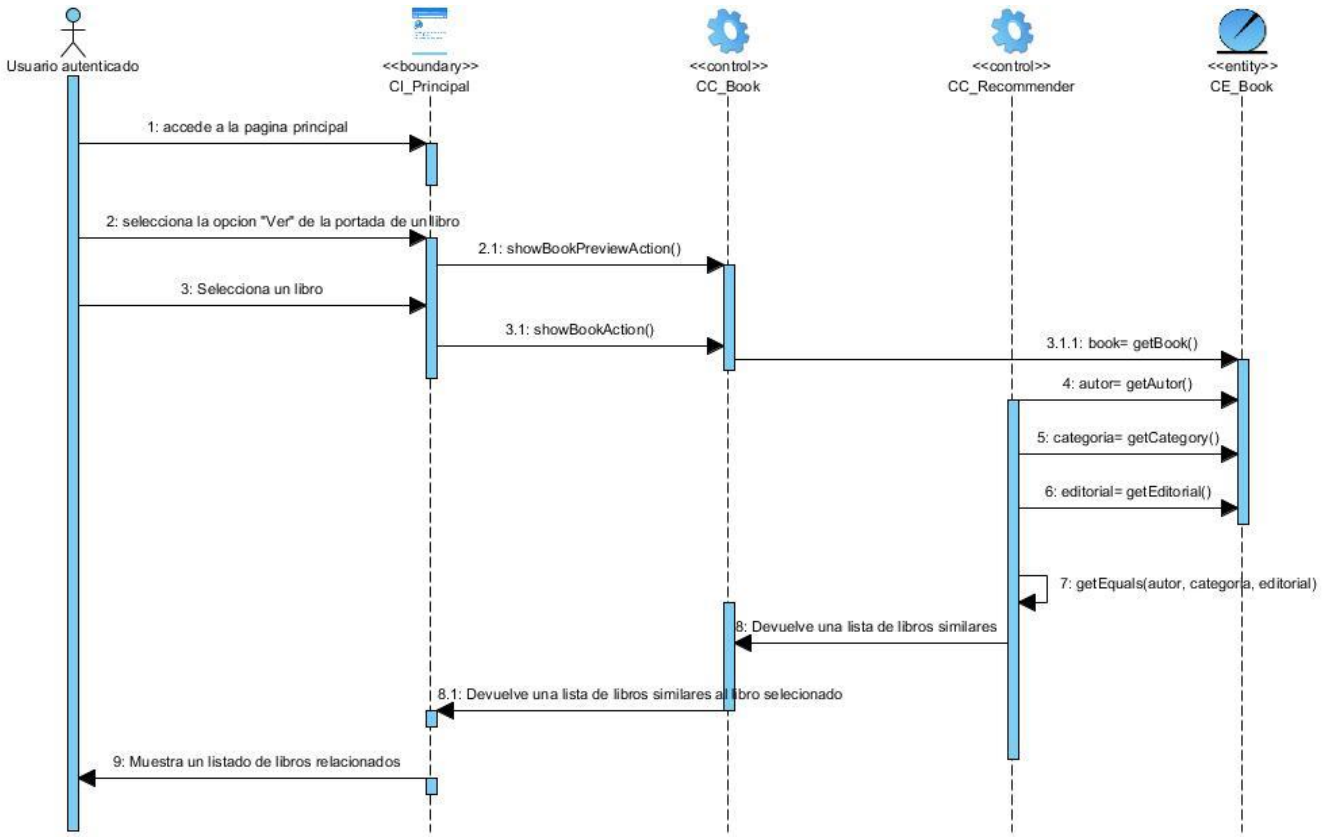


Figura 34: DS Mostrar libros relacionados.

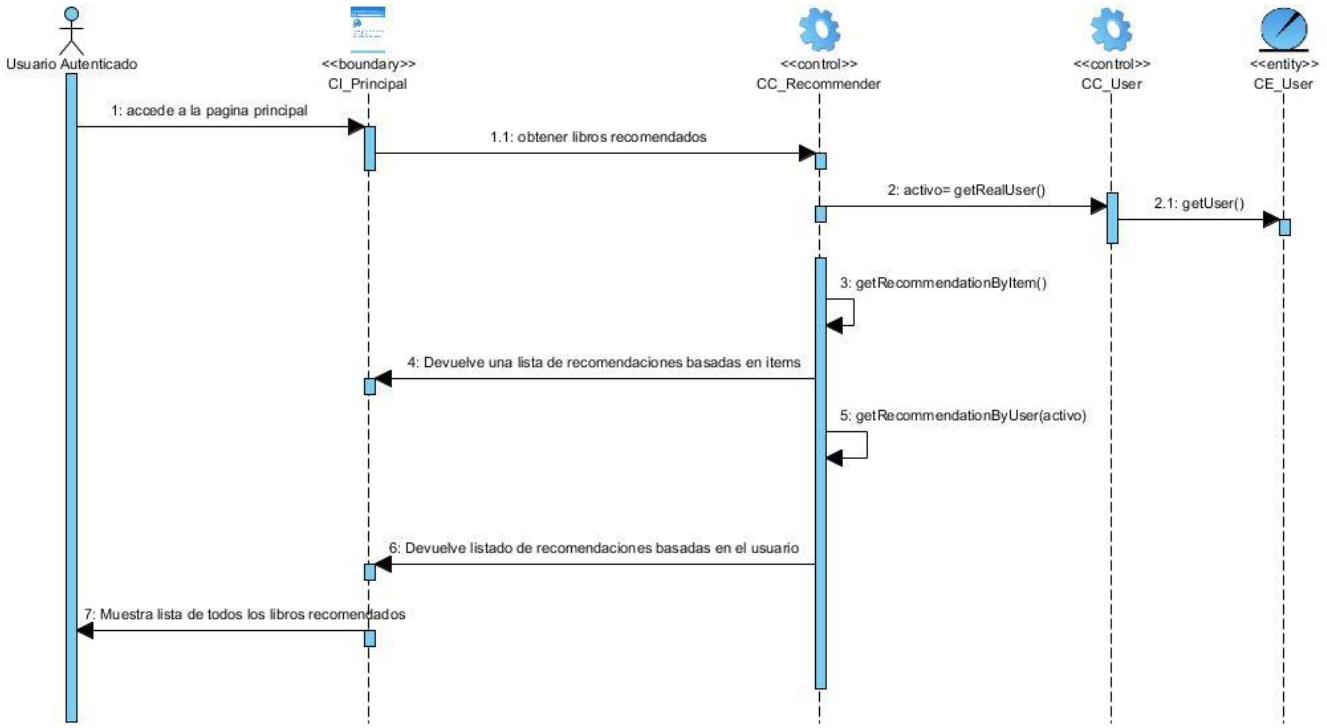


Figura 35: DS Mostrar libros recomendados por el sistema.

Anexo #4: Descripción de las tablas de la Base de Datos

tb_visits_book		
Descripción: Esta tabla es la encargada de almacenar los datos de las visitas a los libros.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Campo que contiene el identificador de la visita.
resource_id	Integer(10)	Campo que contiene el identificador del libro visitado.
user_id	Integer(10)	Campo que contiene el identificador del usuario que realizo la visita.

tb_evaluation_book		
Descripción: Esta tabla es la encargada de almacenar los datos de las evaluación realizadas a los libros.		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Campo que contiene el identificador de la evaluación.
book_id	Integer(10)	Campo que contiene el identificador del libro evaluado.
favorite	boolean	Campo que identifica si el usuario evaluó el libro como favorito o no.
evaluation	Float(8)	Campo que contiene el valor de la evaluación del libro.
user_id	Integer(10)	Campo que contiene el identificador del usuario que realizo la evaluación.

tb_editorial		
Descripción: Esta tabla es la encargada de almacenar los datos de las editoriales		
Atributo	**Tipo**	**Descripción**
id	Integer(10)	Campo que contiene el identificador de la editorial.
name	Varchar(255)	Campo que contiene el nombre de la editorial.
description	text	Campo que contiene la descripción de la editorial.
editorial_email	text	Campo que contiene el correo electrónico de la editorial.
address	text	Campo que contiene la dirección de la editorial.
email	text	Campo que contiene el correo postal de la editorial.
contact_name	text	Campo que contiene el nombre de contacto.
phone	text	Campo que contiene el teléfono de la editorial.
image	Varchar(255)	Campo que contiene la imagen que identifica a la editorial.
is_visible	Boolean	Campo que identifica si la editorial está visible o no.

Anexo #5: Diagrama de componentes (DCOM)

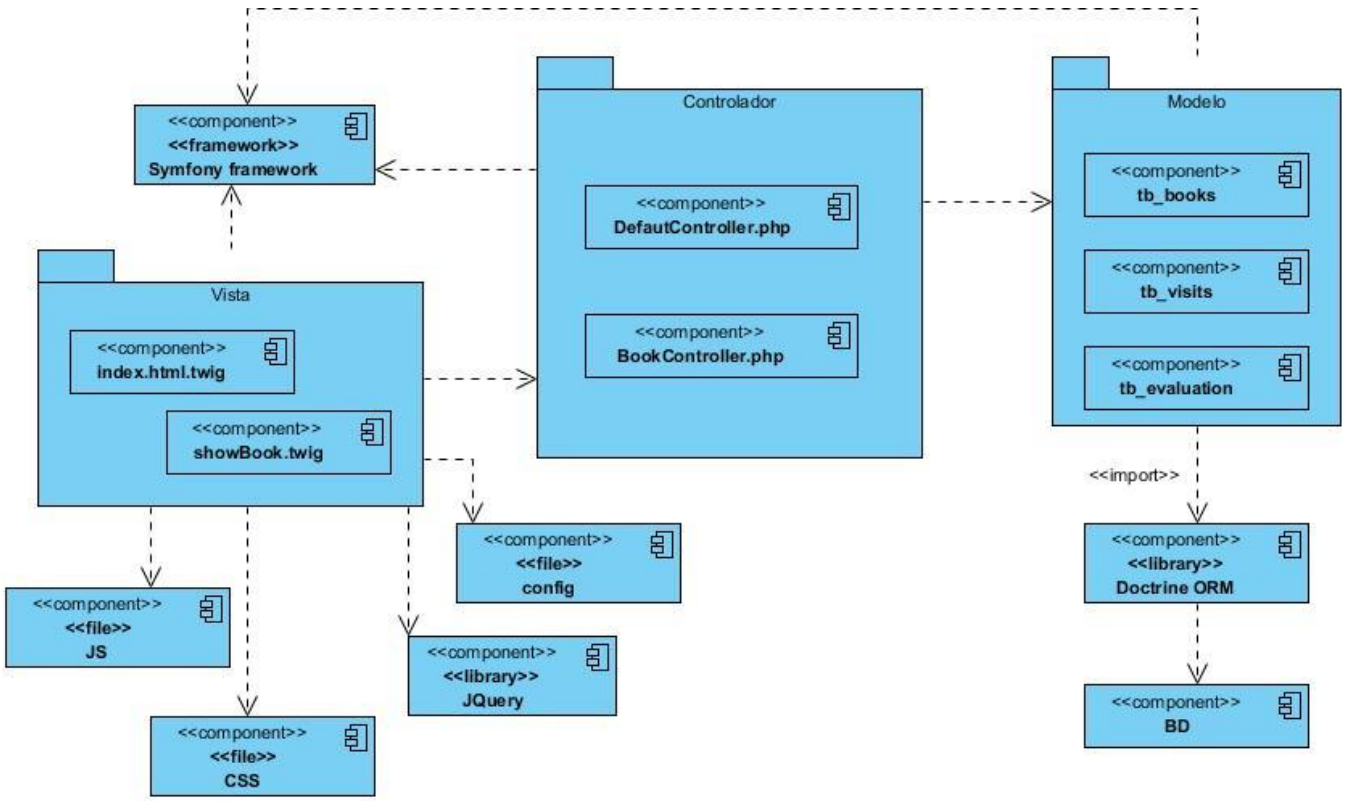


Figura 36: DCOM Crear perfil de usuario.

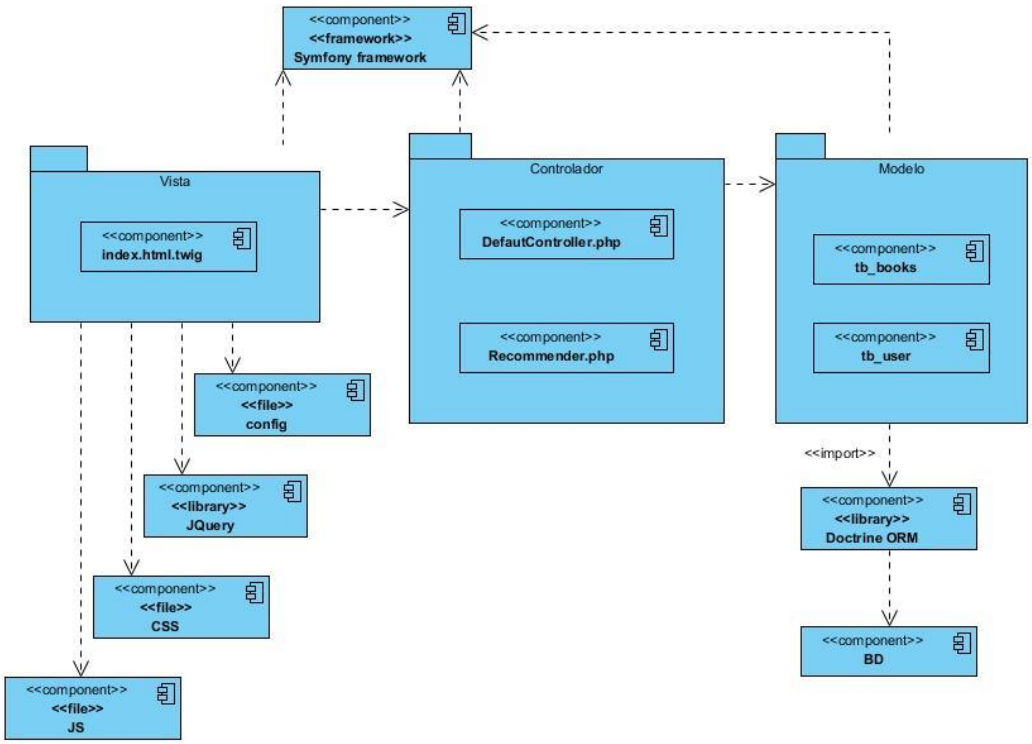


Figura 37: DCOM Mostrar recomendaciones basadas en el usuario.

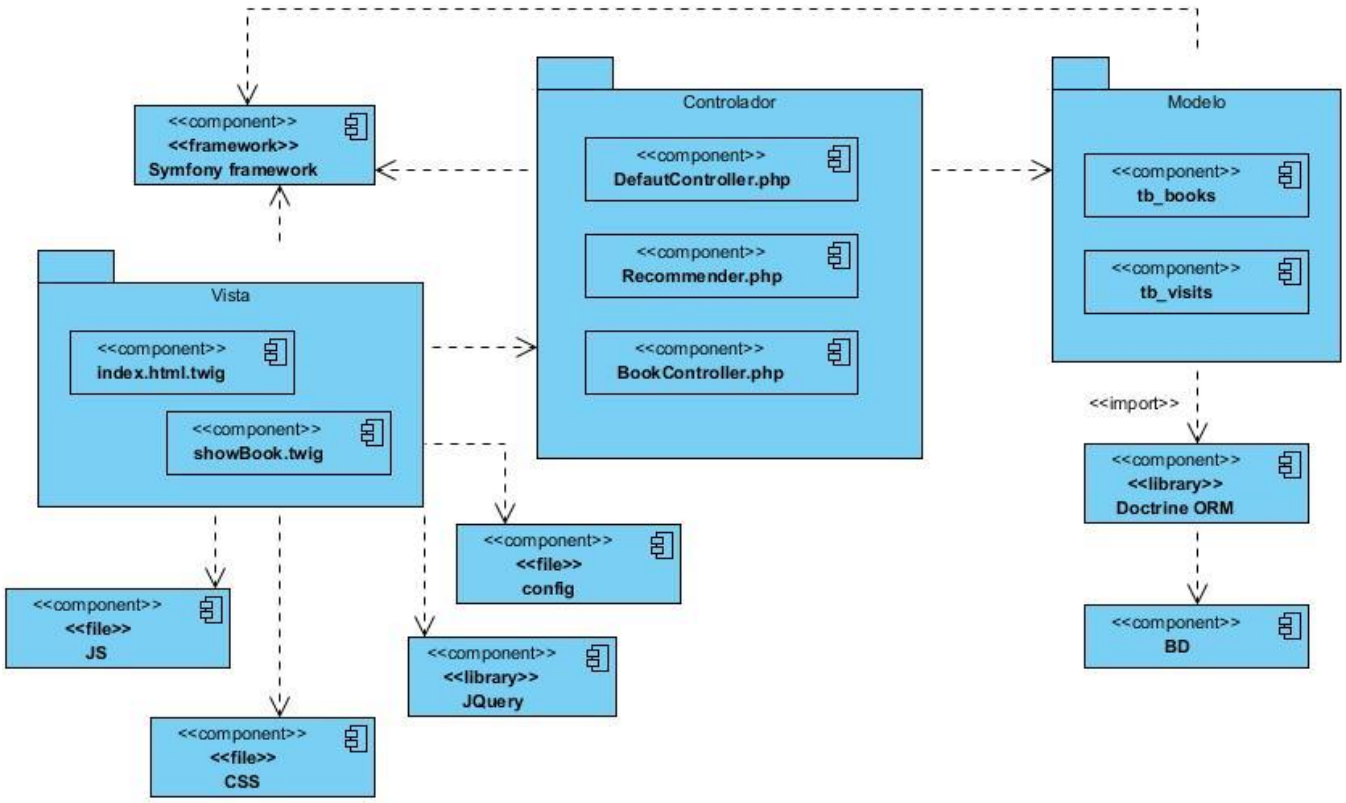


Figura 38: DCOM Mostrar libros relacionados.

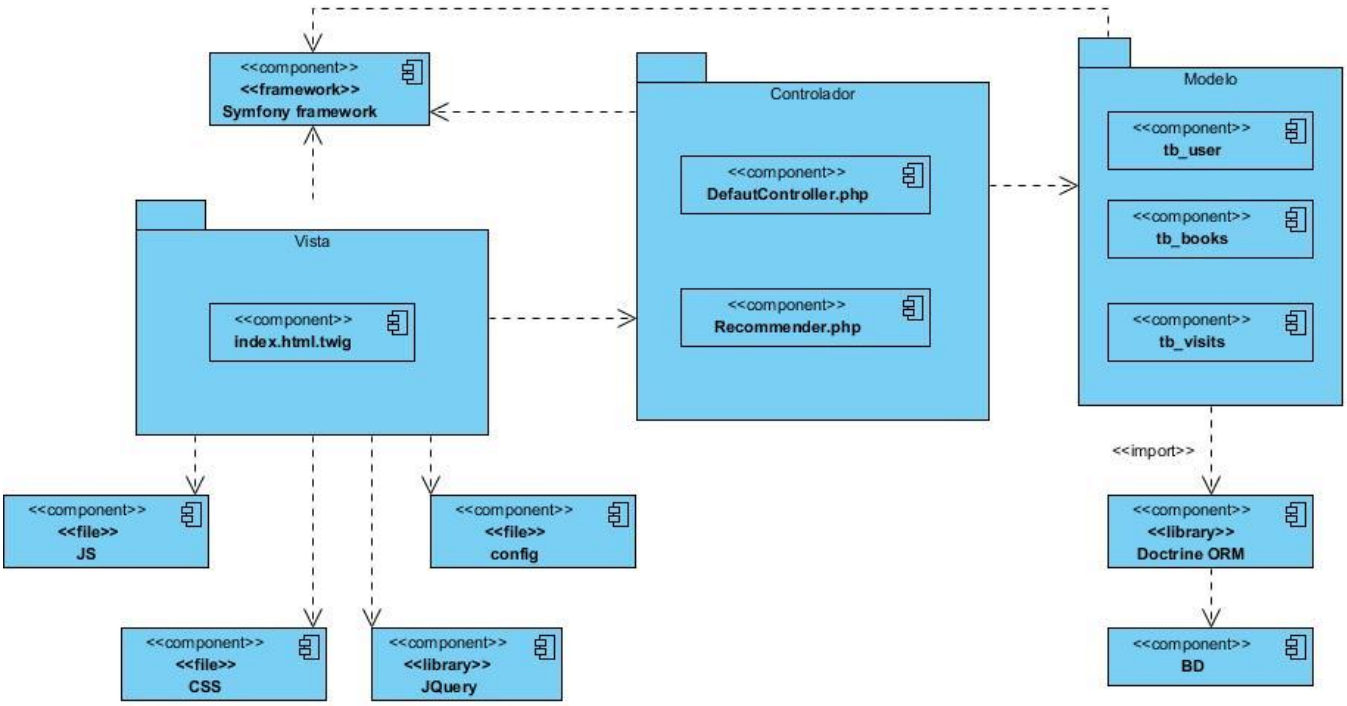


Figura 39: DCOM Mostrar libros recomendados por el sistema.

Anexo #6: Diseño de Casos de Prueba

CP Mostrar Recomendación basada en el usuario			
Descripción: Cuando el usuario accede a la plataforma se le muestra una lista de recomendaciones basada en los usuarios con preferencias similares. **Condiciones de Ejecución:** El usuario debe estar autenticado. El usuario debe haber evaluado al menos un libro.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo básico**
EC 1.1: Selecciona la opción **Entrar**.	Una vez en la portada de la aplicación el usuario selecciona la opción **Entrar** que le permite autenticarse en el sistema.	Muestra una página que contiene un formulario para que el usuario se autentique.	Portada/Login
EC 1.2: Inserta los datos y selecciona la opción **Entrar**.	Una vez insertado el usuario y la contraseña selecciona la opción **Entrar.**	Muestra la portada	Portada/Login/Portada
EC 1.3: Mostrar recomendaciones	Una vez que el usuario se autentica se muestra en la	Muestra un carrusel con una lista de	Portada

basadas en el usuario.	portada un carrusel con una lista de recomendaciones	recomendaciones basadas en los usuarios con preferencias similares.	

CP Mostrar libros relacionados			
Descripción: Se muestra una lista de libros relacionados teniendo en cuenta el autor, editorial y la categoría.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo básico**
EC 1.1: Accede a la plataforma.	El usuario accede a la plataforma.	Se muestra la portada.	Portada
EC 1.2: Selecciona la opción **Ver** de la portada de un libro.	Una vez en la portada el usuario selecciona la opción **Ver** que le permite ver los principales datos del libro y una lista de relacionados.	Se muestra una ventana con los principales datos del libro como: autor, categoría, cantidad de visitas, si es favorito, etc. Además se muestra una lista de libros relacionados.	Portada
EC 1.3: Selecciona la opción **Entrar**.	Una vez en la portada de la aplicación el usuario selecciona la opción **Entrar** que le permite autenticarse en el sistema.	Muestra una página que contiene un formulario para que el usuario se autentique.	Portada/Login
EC 1.4: Inserta los datos y selecciona la opción **Entrar**.	Una vez insertado el usuario y la contraseña selecciona la opción **Entrar.**	Muestra la portada	Portada/Login/Portada
EC 1.5: Selecciona un libro para visitar.	Una vez en la portada el usuario selecciona un libro lo que le permite visitarlo.	Se muestra una ventana con todos los datos del libro y en la parte inferior una serie de pestañas, entre las cuales se encuentra la opción **Libros relacionados.**	Portada/show_book
EC 1.6: Selecciona la pestaña **Libros relacionados.**	Una vez en la visitado el libro el usuario selecciona la pestaña **Libros relacionados.**	Se despliega una lista de libros relacionados con el libro visitado.	Portada/show_book

CP Mostrar libros recomendados por el sistema			
Descripción: Cuando el usuario accede a la plataforma se le muestra una lista de recomendaciones basada en los usuarios con preferencias similares y los libros que ha visitado. **Condiciones de Ejecución:** El usuario debe estar autenticado. El usuario debe haber evaluado al menos un libro.			
Escenario	**Descripción**	**Respuesta del sistema**	**Flujo básico**
EC 1.1: Selecciona la opción **Entrar**.	Una vez en la portada de la aplicación el usuario selecciona la opción **Entrar** que le permite autenticarse en el sistema.	Muestra una página que contiene un formulario para que el usuario se autentique.	Portada/Login
EC 1.2: Inserta los datos y selecciona la opción **Entrar**.	Una vez insertado el usuario y la contraseña selecciona la opción **Entrar.**	Muestra la portada	Portada/Login/Portada
EC 1.3: Mostrar recomendaciones basadas en el usuario y en los libros visitados.	Una vez que el usuario se autentica se muestra en la portada un carrusel con una lista de recomendaciones	Muestra un carrusel con una lista de recomendaciones basadas en los usuarios con preferencias similares y en los libros visitados.	Portada

Anexo #7: Entrevista al cliente y al especialista

Objetivo: Obtener información relacionada con el módulo que se desea desarrollar.

¿En qué consiste el producto que desea desarrollar?

¿A qué tipo de usuarios está destinado este producto?

¿Tiene algún producto creado actualmente?

¿Qué servicios brindará esta aplicación?

¿Cómo los administradores de la aplicación gestionaran la información de los libros requerida?

¿Bajo qué criterio de búsqueda será mostrada y ordenada la información de los libros recomendados?

Algún criterio en especial para lograr un recomendador personalizado adaptable para cualquier usuario.