



Facultad 2

Módulo de recomendación de libros para la biblioteca de la Universidad de Ciencias Informáticas

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

Autor: Ezequiel Brito Navarro

Tutor:

Ing. Wendolyn Rodríguez León

La Habana, septiembre de 2023

Año 65 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título “Módulo de recomendación de libros para la biblioteca de la Universidad de Ciencias Informáticas” concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firman la presente a los 22 días del mes de septiembre del año 2023.

Ezequiel Brito Navarro

Ing. Wendolyn Rodríguez León

Firma del Autor

Firma del Autor

Resumen

El actual avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha tenido un notable impacto en la sociedad, especialmente en el comercio electrónico. En este trabajo de tesis, se presenta el desarrollo de un módulo recomendador de libros para una biblioteca, basado en el filtrado colaborativo tanto por usuario como por ítem, con el propósito de orientar a los usuarios hacia la selección de libros acorde a sus preferencias de lectura, reduciendo así el tiempo dedicado a la búsqueda. Se aborda un análisis de las clasificaciones de sistemas de recomendación, se describen algoritmos y técnicas, y se sigue la metodología AUP-UCI para la implementación, finalizando con la validación de la aplicación para garantizar su correcto funcionamiento.

Palabras claves: Filtrado colaborativo, libros, ítem, usuario

Índice

INTRODUCCIÓN	1
CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.....	5
1.1 SISTEMAS RECOMENDADORES.....	5
1.1.1 <i>Sistemas Recomendadores basados en filtrado colaborativo</i>	5
1.1.2 <i>Sistemas Recomendadores basados en el conocimiento</i>	6
1.1.3 <i>Sistemas Recomendadores basados en contenido</i>	7
1.1.4 <i>Sistemas de recomendación híbridos</i>	7
1.2 ALGORITMOS EN LOS SISTEMAS RECOMENDACIONES	8
1.2.1 <i>Técnicas algorítmicas basadas en memoria</i>	8
1.3 MÓDULO DE RECOMENDACIÓN BIBLIOTECARIO	11
1.4 ANÁLISIS DE SOLUCIONES SIMILARES PARA LA RECOMENDACIÓN DE LIBROS.....	13
1.5 METODOLOGÍA Y MÉTODO DE DESARROLLO.....	16
1.5.1 <i>Metodología de Desarrollo de software</i>	16
1.5.2 <i>Metodología para el trabajo con las técnicas de Aprendizaje Automático</i>	19
1.6 HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS	23
1.6.1 <i>Herramientas de Desarrollo</i>	23
1.6.2 <i>Lenguajes de Desarrollo</i>	25
1.6.3 <i>Sistemas Gestores de Bases de Datos</i>	28
CONCLUSIONES PARCIALES	29
CAPÍTULO 2: ANÁLISIS Y DISEÑO DE LA PROPUESTA DE SOLUCIÓN.....	31
2.1 DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN.....	31
2.2 USUARIOS RELACIONADOS CON EL SISTEMA	31
2.3 MODELO DE DOMINIO	32
2.4 FUNDAMENTACIÓN DEL TIPO SISTEMA RECOMENDADOR SELECCIONADO	33
2.5 ESPECIFICACIÓN DE REQUISITOS	37
2.5.1 <i>Requisitos Funcionales</i>	37
2.5.2 <i>Requisitos No Funcionales</i>	39
2.6 HISTORIAS DE USUARIO	40
2.7 ARQUITECTURA DEL SISTEMA.....	41
2.7.1 <i>Patrón Arquitectónico</i>	41
2.7.2 <i>Patrones de Diseño</i>	43
2.7.3 <i>Implementación del algoritmo siguiendo las fases de KDD</i>	45
2.7.3.1 Selección de los datos.....	45
2.7.3.2 Minería de datos.....	46
CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA DE LA PROPUESTA DE SOLUCIÓN.....	49
3.1 DIAGRAMA DE DESPLIEGUE	49
3.2 ESTRATEGIA DE PRUEBAS DE SOFTWARE.....	50
3.2.1 <i>Pruebas Unitarias</i>	51
3.2.2 <i>Pruebas de precisión</i>	52
3.2.3 <i>Pruebas de aceptación</i>	54
CONCLUSIONES PARCIALES	57
CONCLUSIONES GENERALES.....	58

RECOMENDACIONES	59
REFERENCIAS BIBLIOGRÁFICAS.....	60
ANEXOS	62
ANEXO 1 HISTORIAS DE USUARIO	62

Índice de Tablas

Tabla 1 Comparación de soluciones homólogas.....	15
Tabla 2 Comparación de las metodologías	19
Tabla 3 Usuarios del sistema.....	31
Tabla 4 Requisitos funcionales.....	38
Tabla 5 Requisitos no funcionales	39
Tabla 6 HU Mostrar Recomendaciones basadas en ítem	41
Tabla 7 Caso de prueba de aceptación #1	55
Tabla 8 Caso de prueba de aceptación #2.....	56
Tabla 9 Caso de prueba de aceptación #3.....	56
Tabla 10 HU Calcular índice	62
Tabla 11 HU Usuarios Semejantes.....	62
Tabla 12 HU Ítems Semejantes.....	63
Tabla 13 HU Predicción por usuario.....	63
Tabla 14 HU Recomendaciones basadas en usuario	64
Tabla 15 HU Recomendaciones basadas en ítem.....	65
Tabla 16 HU Mostrar Recomendaciones basadas en usuario.....	65
Tabla 17 HU Mostrar Libros relacionados	66
Tabla 18 HU Mostrar Libros recomendados	67

Índice de Figuras

Figura 1 Pasos a seguir en el uso de la técnica KNN	10
Figura 2 Etapas de KDD	21
Figura 3 Modelo de dominio	33
Figura 4 Matriz de predicciones	37
Figura 5 Recomendación de los n ítems de todos los posibles	37
Figura 6 Representación del patrón arquitectónico	43
Figura 7 Fragmento de código de las url.....	44
Figura 8 Fragmento de código del archivo models.py	45
Figura 9 Modelo de datos	46
Figura 10 Diagrama de despliegue	49
Figura 11 Fragmento de código del archivo unittest	52
Figura 12 Fragmento de código de prueba de precisión	54
Figura 13 No conformidades detectadas.....	57

Introducción

En la actualidad, el vertiginoso avance de las Tecnologías de la Información y la Comunicación (TIC) ha revolucionado la forma en que interactuamos con el mundo que nos rodea. Las TIC han penetrado profundamente en la vida cotidiana, transformando no solo la manera en que nos comunicamos, sino también la forma en que trabajamos, estudiamos y accedemos a la información. Esta omnipresente influencia de las TIC ha dado forma a una nueva era en la que la tecnología se ha convertido en el motor impulsor del progreso y la innovación. La irrupción de las TIC en la sociedad contemporánea ha desencadenado una transformación sin precedentes en diversos sectores, y el ámbito bibliotecario no ha quedado al margen de esta revolución tecnológica.

Las TIC, que abarcan una amplia gama de herramientas y sistemas digitales destinados a la gestión, acceso y difusión de información, han revolucionado la forma en que las bibliotecas operan y sirven a sus usuarios. El sector bibliotecario, tradicionalmente asociado con la conservación y el acceso a recursos impresos, se ha visto impulsado a evolucionar y adaptarse a un entorno caracterizado por la creciente digitalización y la demanda de servicios en línea. Estas tecnologías han abierto nuevas perspectivas y posibilidades para mejorar la eficiencia, accesibilidad y alcance de las bibliotecas en la era digital.

Cuba, una isla ubicada en el corazón del Caribe, se destaca no solo por su rica historia y cultura, sino también por su compromiso con la educación y el acceso a la información. Las bibliotecas en Cuba ofrecen una variedad de servicios que van desde la conservación del patrimonio cultural hasta la facilitación de investigaciones académicas y la promoción de la lectura recreativa. Además, el país ha enfrentado desafíos y limitaciones debido a factores económicos y políticos, lo que ha influido en la disponibilidad de recursos y la adopción de tecnologías modernas en el sector bibliotecario.

En el marco de las iniciativas impulsadas por el gobierno cubano para promover la utilización de las Tecnologías de la Información y Comunicación (TIC), surgió la Universidad de Ciencias Informáticas (UCI) como un proyecto emblemático. Esta institución, fundada en 2002 en el espacio previamente ocupado por la Base Lourdes, tenía como objetivo principal elevar el nivel cultural de la población y mejorar la calidad de vida a través del desarrollo de competencias en el campo de la informática y la tecnología. La UCI se posiciona como un referente en la convergencia entre la tecnología y la educación superior. Esta institución, dedicada a la formación de profesionales en el campo de la informática, ha

reconocido la importancia de contar con una biblioteca actualizada y eficiente para apoyar las necesidades de su comunidad académica.

En el corazón de este centro de excelencia académica, se encuentra su biblioteca, una pieza fundamental en el engranaje educativo de la UCI. La biblioteca en el campus universitario desempeña un papel crucial como centro de conocimiento y recurso esencial para estudiantes, profesores e investigadores. Este espacio se caracteriza por albergar una vasta colección de materiales bibliográficos, tanto impresos como digitales, que abarcan una amplia gama de disciplinas informáticas y tecnológicas. La biblioteca en la UCI no solo se destaca por su extenso acervo de libros, revistas y recursos digitales, sino también por su compromiso con la innovación y la incorporación de tecnologías modernas para facilitar el acceso a la información. Sin embargo, a pesar de sus recursos y servicios de vanguardia, la biblioteca de la UCI se enfrenta a desafíos específicos, ya que cuenta con un portal web bibliotecario, que carece de eficiencia en cuanto a la recomendación de nuevos libros a los usuarios en base a sus gustos literarios e intereses específicos. Esto ha generado una falta de personalización en la experiencia de los estudiantes y profesores al momento de buscar nuevos materiales para su lectura. Actualmente, los usuarios deben revisar manualmente los títulos disponibles, provocando que se empleen muchas horas para dicha tarea.

Partiendo de la situación problemática expuesta anteriormente se plantea el siguiente **problema de investigación**: ¿Cómo mejorar la experiencia de los usuarios que visitan el portal web de la biblioteca de la UCI en el proceso de acceder a nuevas temáticas de interés?

Objeto de Estudio: Portales web de instituciones bibliotecarias.

Para dar solución al problema de investigación planteado, se define como **Objetivo General**: Desarrollar un módulo de recomendación de libros que brinde la funcionalidad de sugerir literatura acorde a los intereses y preferencias individuales de cada usuario.

Campo de Acción: Módulos recomendadores de libros.

Objetivos Específicos:

- Fundamentar de los referentes teóricos-metodológicos relacionados con la recomendación de literatura en base a intereses individuales.

- Diseñar un módulo de recomendación literaria que se adapte al portal web de la UCI, respetando todos los patrones de diseño definidos propios del portal.
- Implementar el módulo de recomendación teniendo en cuenta los artefactos obtenidos del diseño.
- Validar el sistema propuesto a partir de los métodos definidos en la investigación.

Método Científico

El método científico de investigación es la forma de abordar la realidad, de estudiar la naturaleza, la sociedad y el pensamiento, con el propósito de descubrir su esencia y sus relaciones. El método científico se puede clasificar en teóricos y empíricos, los cuales están dialécticamente relacionados.

Métodos Teóricos

- **Analítico-Sintético:** Permite realizar un análisis de teorías y documentos en relación con el tema en cuestión, y con ello realizar un estudio de las características y necesidades de acuerdo a la solución de la investigación, para de esta forma establecer las bases teóricas en relación con el desarrollo del módulo.
- **Histórico-Lógico:** Para realizar un estudio de la trayectoria histórica, evolución y desarrollo del portal web.

Métodos empíricos

- **Análisis documental:** Mediante la selección y recolección de la documentación asociada al portal web, se hace uso de este método con el objetivo de extraer la información necesaria para lograr los resultados esperados.
- **Modelado:** Se utiliza en la modelación de los diagramas correspondientes a la etapa de análisis, diseño e implementación del módulo a desarrollar en la presente investigación.

La presente investigación consta de tres (3) capítulos, los cuales se resumen brevemente a continuación:

Capítulo 1: Fundamentación Teórica

En este capítulo se expone los conceptos relacionados con el dominio de la investigación y se realiza un análisis de la teoría para el desarrollo del módulo. Además, se analizan algunas herramientas, metodologías y lenguaje de programación, así como el entorno de desarrollo que se utilizará para dar solución al problema planteado en la investigación.

Capítulo 2: Análisis y Diseño de la propuesta

En este capítulo se realiza el análisis y diseño de la propuesta de solución según los requisitos funcionales obtenidos. Se realiza una descripción del sistema que se pretende desarrollar, se elaboran las Historias de Usuarios para identificar las funcionalidades con las que debe cumplir el sistema, además de presentar la arquitectura seleccionada para su desarrollo.

Capítulo 3: Implementación y prueba de la propuesta

En este capítulo se describen los estándares de codificación empleados en la implementación de la propuesta de solución. Se describe la estrategia de prueba definida para validar la propuesta y se realiza un análisis de los resultados obtenidos luego de aplicadas las mismas.

Capítulo 1: Fundamentación Teórica

En el presente capítulo se esboza el marco teórico, donde se abordan los principales conceptos y definiciones necesarios para el desarrollo de la investigación. Se describen los sistemas existentes que responden al objeto de estudio y campo de acción de la investigación, ya sea en el escenario nacional o internacional, realizando posteriormente una valoración de los mismos. Además, se explica la selección de la metodología de desarrollo de software y se fundamenta mediante un estudio los lenguajes, herramientas y tecnologías utilizadas para el desarrollo de la propuesta de solución.

1.1 Sistemas Recomendadores

Según Ricci, los sistemas recomendadores “son herramientas de software y técnicas que ofrecen determinadas sugerencias de empleo y utilidad para el usuario”. Un sistema de recomendación es aquel sistema que tiene como principal tarea seleccionar ciertos objetos de acuerdo a los requerimientos del usuario. Constituyen una técnica de filtrado de información, la cual presenta distintos tipos de temas o ítems de información al usuario, basándose en la predicción del “ranking” o ponderación que este le daría a un ítem que el sistema aún no ha considerado (Ricci et al., 2011); mediando, automatizando o soportando de esta forma, el proceso de realizar recomendaciones, además de utilizarse para predecir la valoración de los usuarios sobre elementos que aún no han evaluado (Rensick, 2004).

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. Las clasificaciones más aceptadas y reconocidas por los principales autores en el tema son los que a continuación se exponen:

1.1.1 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 (Peña, 2008).

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 (Cingolani, 2014a).

A continuación se explican los tipos de filtrado de información según esta técnica de recomendación:

- **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 (Peña, 2008).

1.1.2 Sistemas Recomendadores basados en el conocimiento

Según (Cingolani, 2014b), 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.

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) (Cingolani, 2014a).

1.1.3 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 esta 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 esta técnica es que los sistemas ofrecen ítems similares a los vistos con anterioridad, lo que significa que no introducen ninguna novedad (Escudero, 2014).

1.1.4 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 (Escudero, 2014).

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) (Peña, 2008).

1.2 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 (Capouya, 2009). 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:

- **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) (Moya, 2013).

1.2.1 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 (Velez-Langs & Santos, 2006).

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 u_i y u_j que pertenecen a R y se aplican algunos de los siguientes métodos:

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

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

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 (Vera et al., 2015). A continuación se muestra el funcionamiento del proceso de recomendación utilizando la técnica KNN.

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 1):

- **Cálculo de similaridad entre usuarios:** De acuerdo a una serie de técnicas se procede a calcular la similaridad entre diferentes usuarios.
- **Calcular los K-Vecinos:** A partir de los datos obtenidos con las técnicas se obtienen los usuarios con mayor grado de similaridad.
- **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.
- **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 (Riveros Rey et al., 2017).

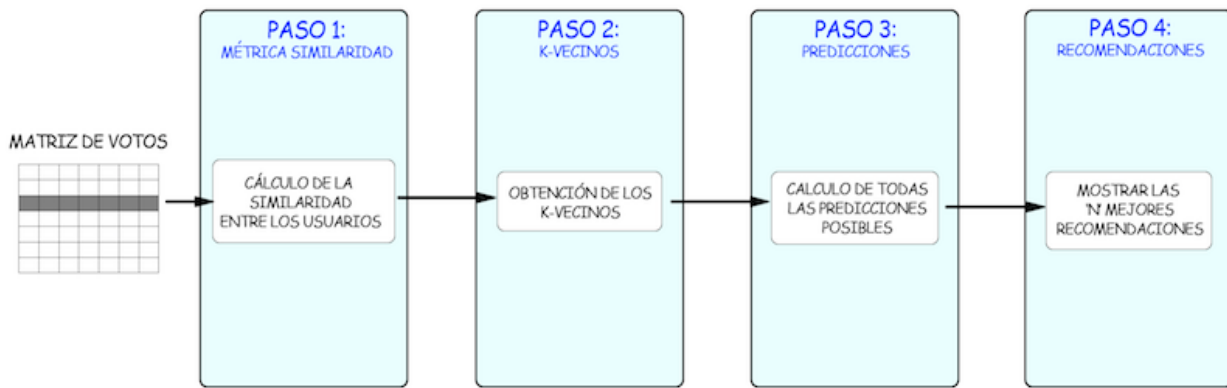


Figura 1 Pasos a seguir en el uso de la técnica KNN

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 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 (Moya, 2013).

- **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) (Velez-Langs & Santos, 2006).

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

Ecuación 2 Correlación de Pearson

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.

r_{ij} : Índice de preferencia del usuario i con el ítem j .

r_{jk} : 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 3. 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 (Peña, 2008).

$$\text{sim}_{\text{Cos}}(A, B) = \frac{\vec{A} * \vec{B}}{|\vec{A}| * |\vec{B}|}$$

Ecuación 3 Similitud Basada en Coseno

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 uno de los primeros algoritmos que se usan en los SR basados en filtrado colaborativo.

1.3 Módulo de recomendación bibliotecario

Los modelos de recomendación bibliotecarios son herramientas esenciales para mejorar la experiencia de los usuarios en las bibliotecas al proporcionarles recomendaciones personalizadas de libros y otros recursos de lectura. El objetivo central de estos modelos es mejorar la experiencia de los usuarios al

facilitarles la búsqueda y selección de libros que se ajusten a sus preferencias de lectura. Utilizan técnicas avanzadas de análisis de datos y aprendizaje automático para cumplir con este objetivo y, al mismo tiempo, se adaptan a medida que los gustos de los usuarios evolucionan con el tiempo. Estas herramientas son fundamentales para promover la lectura y garantizar que los recursos de la biblioteca se utilicen de manera efectiva.

Características:

Los modelos de recomendación bibliotecarios se basan en la personalización. Utilizan algoritmos que analizan el comportamiento de lectura pasado de los usuarios, sus preferencias declaradas y otros datos relevantes para crear perfiles de lectura individuales. Esto permite sugerir libros que sean relevantes para cada usuario en particular. Aunque la personalización es fundamental, los modelos de recomendación también pueden incorporar la diversidad en sus sugerencias. Esto significa que no solo recomiendan libros similares a los que el usuario ya ha leído, sino que también sugieren libros que podrían ampliar sus horizontes de lectura. Estos modelos suelen actualizarse regularmente para reflejar los cambios en los gustos y preferencias de los usuarios. A medida que un usuario lee nuevos libros o cambia sus intereses, el sistema adapta las recomendaciones en consecuencia.

Las principales funciones de un módulo de recomendación bibliotecario incluyen:

- **Perfil de usuario:** Crear un perfil de lectura para cada usuario, basado en su historial de búsqueda, préstamos anteriores y otros datos relevantes. Esto puede incluir géneros preferidos, autores favoritos y temas de interés.
- **Análisis de contenido:** Analizar la información bibliográfica de los libros en la colección de la biblioteca, como el título, el autor, la temática y las etiquetas, para comprender su contenido y características.
- **Comparación de perfiles:** Comparar el perfil de lectura del usuario con las características de los libros en la colección para encontrar coincidencias y determinar qué libros son más adecuados para ese usuario en particular.
- **Generación de recomendaciones:** Utilizar algoritmos de recomendación para generar una lista de libros recomendados para el usuario. Estas recomendaciones suelen basarse en la similitud entre el perfil del usuario y las características de los libros.

- **Interfaz de usuario:** Presentar las recomendaciones de una manera accesible y atractiva a través de la interfaz en línea de la biblioteca, lo que permite a los usuarios explorar y seleccionar los libros recomendados.
- **Retroalimentación:** Recopilar retroalimentación de los usuarios sobre las recomendaciones, como si les gustaron o no, si los leyeron o no, etc., para mejorar continuamente el sistema y afinar las sugerencias futuras (Parra Valero, 2016).

1.4 Análisis de soluciones similares para la recomendación de libros

Existen varios módulos de recomendación de libros para bibliotecas a nivel internacional, cada uno con sus propias características y funcionalidades. A continuación, se describen brevemente tres de los sistemas más destacados:

LibraryThing for Libraries:

LibraryThing es un servicio en línea para ayudar a las personas a catalogar sus libros fácilmente. Puedes acceder a tu catálogo desde cualquier lugar, incluso desde tu teléfono móvil. Debido a que todos catalogan juntos, LibraryThing también conecta a las personas con los mismos libros, se le ocurren sugerencias sobre qué leer a continuación, etc (Charles, 2020).

Características:

- Ofrece recomendaciones de libros basadas en etiquetas (tags) y perfiles de lectura de usuarios.
- Permite a los usuarios crear listas de lectura y ver recomendaciones similares.
- Proporciona reseñas de libros escritas por la comunidad de usuarios.
- Integra con el catálogo de la biblioteca y puede sugerir libros relacionados durante las búsquedas.

BiblioCommons:

BiblioCommons es un servicio en línea para hacer reservaciones, renovaciones y buscar títulos en la colección de la Biblioteca Pública de Oakland. BiblioCommons también te permite compartir

calificaciones y comentarios sobre los títulos que encuentra en su biblioteca y conectarse con otros usuarios de la biblioteca (Santos, 2021).

Características:

- Proporciona recomendaciones de libros basadas en el historial de préstamos de los usuarios y sus interacciones en el catálogo de la biblioteca.
- Permite a los usuarios calificar y revisar libros.
- Ofrece listas de lectura temáticas y listas populares creadas por otros usuarios.
- Facilita la creación y gestión de listas de lectura personalizadas.

Goodreads

Es una red social o comunidad específica de lectores de libros y que gira en torno a los libros. En GoodReads es posible ver qué libros leen nuestros amigos, cuáles han leído y cuáles quieren leer, valoraciones y críticas de libros, así como conversar con otros lectores, entre otras opciones. Actualmente es la mayor red social de lectores y autores y de recomendaciones de libros (Olah, 2018).

Goodreads actualmente cuenta con 40 millones de miembros, 1.000 millones de libros registrados y 43 millones de valoraciones (reviews). La aplicación nace en 2007, pero desde 2013 Goodreads pertenece a Amazon cuando fue comprada por Jeff Bezos. Cuenta con casi dos millones de seguidores en Twitter y, además, como dato interesante, la mitad de los usuarios de esta red social accede desde móvil (Orenes, 2018).

Características:

- Permite a los usuarios calificar y revisar libros.
- Genera recomendaciones personalizadas en función de las calificaciones y reseñas de los usuarios.
- Ofrece listas temáticas y de género creadas por la comunidad de usuarios.
- Facilita la creación y seguimiento de listas de lectura personalizadas.

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

Tabla 1 Comparación de soluciones homólogas

Funcionalidad	BiblioCommons	LibraryThing for Libraries	Goodreads
Recomendaciones personalizadas basadas en historial de préstamos y comportamiento de usuarios	Sí	Sí	Sí
Calificaciones y reseñas de usuarios	Sí	No	Sí
Listas de lectura temáticas creadas por usuarios	Sí	Sí	Sí
Resúmenes y reseñas de libros	Sí	Sí	Sí
Búsqueda por etiquetas (tags)	No	Sí	Sí
Creación y gestión de listas de lectura personalizadas	Sí	No	Sí
Integración con el catálogo de la biblioteca	Sí	Sí	No
Comunidad de usuarios activa	Sí	Sí	Sí

Luego del análisis anterior, se llega a la conclusión de que es esencial incorporar la capacidad de proporcionar recomendaciones personalizadas basadas en el historial de préstamos y el comportamiento de los usuarios, lo que asegura que las sugerencias sean pertinentes a sus intereses. La opción de calificar y revisar libros permite a los usuarios contribuir con sus opiniones y retroalimentar el sistema. Las listas de lectura temáticas creadas por la comunidad y las herramientas para crear y

administrar listas de lectura personalizadas empoderan a los usuarios y fomentan su participación activa. Además, la inclusión de resúmenes y reseñas de libros enriquece la información sobre las recomendaciones, facilitando decisiones informadas. La capacidad de búsqueda por etiquetas y la integración fluida con el catálogo de la biblioteca complementarán la experiencia del usuario, asegurando que puedan acceder rápidamente a los libros recomendados. Al combinar estas funcionalidades, el módulo de recomendación de libros proporcionará una experiencia completa y personalizada, mejorando la satisfacción de los usuarios y fomentando la exploración y el descubrimiento de nuevos materiales de lectura en tu biblioteca.

1.5 Metodología y método de desarrollo

1.5.1 Metodología de Desarrollo de software

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema de informático. Suele estar documentada y promovida por algún tipo de organización ya sea esta pública o privada. Tienen como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, exigiéndose así que el proceso sea configurable. Las metodologías de desarrollo se clasifican en dos clases: las metodologías tradicionales o robustas y las ágiles o ligeras (Pressman, 2014).

Las metodologías se pueden clasificar en dos grupos. Las metodologías tradicionales, que buscan siempre una fuerte planificación y documentación durante todo el desarrollo, y las metodologías ágiles, en las que se enfoca al desarrollo de software el cual es incremental, cooperativo, sencillo y adaptado. Para el desarrollo de la solución es imprescindible el uso de una metodología de desarrollo de software que controle cada uno de los procesos que lleva a cabo el equipo de trabajo. Teniendo en cuenta que el equipo es pequeño, integrado en su totalidad por dos personas (el desarrollador y el cliente, quien participa constantemente con el equipo de desarrollo), los requerimientos están sujetos a cambios, el proyecto es pequeño y de corto plazo, se utilizará la metodología definida por el centro CIDI de la

Universidad para el desarrollo de proyectos. La misma consiste en una versión desarrollada por la UCI basada en la metodología de desarrollo de software AUP.

Metodología AUP Versión UCI

La metodología utilizada en la universidad es una variante de AUP (Agile Unified Process, Proceso Ágil Unificado). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software, usando técnicas ágiles. Se adapta al ciclo de vida definido para la actividad productiva de la institución, logra estandarizar los productos de trabajo elaborados en los diferentes centros de desarrollo de la universidad (Rodríguez Sánchez, 2015).

AUP UCI establece tres fases:

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

AUP propone siete disciplinas, las cuales son: Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno. Se decide para el ciclo de vida de los proyectos de la UCI mantener las mismas disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajo: Modelado de negocio, Requisitos, Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagregan en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con la áreas de procesos que define CMMI-DEV v1.3 para que pueda emplearse en los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI) para el

nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto) (Rodríguez Sánchez, 2015).

Descripción de los escenarios:

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

- **Escenario 1:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que puedan modelar una serie de interacciones entre los trabajadores del negocio/actores del sistema (usuario), similar a una llamada y respuesta respectivamente, donde la atención se centra en cómo el usuario va a utilizar el sistema.
- **Escenario 2:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio.
- **Escenario 3:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio con procesos muy complejos, independientes de las personas que los manejan y ejecutan, proporcionando objetividad, solidez, y su continuidad.
- **Escenario 4:** Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos (Rodríguez Sánchez, 2015).

Dentro de la metodología se seleccionó el escenario número cuatro, la decisión se adopta en que la propuesta no modela negocio ya que solo se centraría en llevar toda una información de forma digital, se aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. Están centradas en el usuario, tienen una agilidad en el proceso de desarrollo, es una metodología híbrida, es adaptable a los cambios y no genera muchos artefactos para optimizar el tiempo.

1.5.2 Metodología para el trabajo con las técnicas de Aprendizaje Automático

El proceso de extraer conocimiento a partir de grandes volúmenes de datos ha sido reconocido por muchos investigadores como un tópico de investigación clave en los sistemas de bases de datos, y por muchas compañías industriales como una importante área y una oportunidad para obtener mayores ganancias (Timarán, 2009).

El siguiente análisis se centra en cuatro metodologías ampliamente reconocidas en la minería de datos: KDD, CRISP-DM, SEMMA y CATALYST. Cada una de estas metodologías se ha desarrollado para abordar aspectos específicos del ciclo de vida de la minería de datos. A continuación, se presenta una tabla comparativa que destaca diversas fases de estas metodologías, desde la comprensión del negocio hasta la implementación del nuevo conocimiento, con el objetivo de proporcionar una visión detallada de sus enfoques respectivos y facilitar la elección informada de la metodología más adecuada para un proyecto de minería de datos.

Tabla 2 Comparación de las metodologías

Fases	KDD	CRISP-DM	SEMMA	CATALYST
Análisis y comprensión del negocio	Comprensión del dominio de aplicación	Compresión del negocio		Modelado de negocio
Selección y preparación de los datos	Crear el conjunto de datos	Entendimiento de los datos	Muestreo	Preparación de los datos
	Limpieza y pre procesamiento de los datos	Preparación de los datos	Comprensión	
	Reducción y proyección de datos		Modificación	
Modelado	Determinar la tarea de minería	Modelado	Modelado	Selección de herramientas y modelado inicial
	Determinar el algoritmo de minería			
	Minería de datos			

Evaluación	Interpretación	Evaluación	Valoración	Refinamiento del modelo
Implementación	Utilización del nuevo conocimiento	Despliegue		Comunicación

Tras analizar las metodologías anteriores, la elección de la metodología KDD, surge como una decisión fundamentada en la búsqueda de un enfoque integral y versátil. KDD, al abarcar todo el proceso de descubrimiento de conocimiento, desde la comprensión del negocio hasta la evaluación de resultados, proporciona una estructura organizada que simplifica la ejecución del proyecto.

La elección de la metodología KDD para el trabajo con las técnicas de aprendizaje automático en el desarrollo del módulo de recomendación se debe a que es una de las metodologías más adecuadas para el análisis de grandes conjuntos de datos y la detección de patrones y relaciones útiles en los datos. Combina técnicas de análisis de datos tradicionales con tecnologías avanzadas de aprendizaje automático. Además, KDD es un enfoque estructurado y sistemático que puede ayudar a asegurar que todos los pasos del análisis se realicen de manera completa y consistente.

Descubrimiento de conocimiento en bases de datos (KDD)

El Descubrimiento de conocimiento en bases de datos (KDD, del inglés Knowledge Discovery in Databases) es básicamente un proceso automático en el que se combinan descubrimiento y análisis. El proceso consiste en extraer patrones en forma de reglas o funciones, a partir de los datos, para que el usuario los analice. Esta tarea implica generalmente preprocesar los datos, hacer minería de datos (data mining) y presentar resultados (Timarán-Pereira et al., 2016).

En el campo de la minería de datos, varias metodologías son ampliamente conocidas y utilizadas. A continuación, se presentan algunas de las más destacadas:

1. **KDD (Knowledge Discovery in Databases):** KDD no es solo una metodología, sino un proceso que engloba la minería de datos. Incluye diversas etapas, como la selección y preprocesamiento de datos, transformación, minería de datos y evaluación. KDD se centra en la extracción de conocimiento útil a partir de grandes conjuntos de datos.
2. **CRISP-DM (Cross-Industry Standard Process for Data Mining):** Es una metodología estándar que proporciona un enfoque estructurado y detallado para el proceso de minería de

datos. Se organiza en seis fases: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación e implementación.

3. **SEMMA (Sample, Explore, Modify, Model, Assess)**: Es una metodología desarrollada por SAS Institute¹ y se centra en cinco fases: muestreo (*Sample*), exploración (*Explore*), modificación (*Modify*), modelado (*Model*) y evaluación (*Assess*). Es especialmente conocida en el entorno del software SAS.
4. **CATALYST**: Es otra metodología utilizada en minería de datos, que aborda aspectos como la determinación de la tarea de minería, la selección de herramientas y el modelado inicial, la evaluación e interpretación, y finalmente, la implementación y comunicación de los resultados.

El modelo KDD constituyó el primero en ser aceptado en la comunidad científica que estableció las etapas principales de un proyecto de explotación de información. En este modelo se identifican patrones válidos, novedosos, potencialmente útiles y principalmente entendibles, implica el análisis de grandes cantidades de datos observacionales, para encontrar relaciones insospechadas (Moine, 2013). En el modelo KDD es posible definir al menos 6 estados: Recolección de datos, Selección, Limpieza y Transformación de datos, Minería de datos, Evaluación y Validación, Interpretación y Difusión, Actualización y Monitorización.

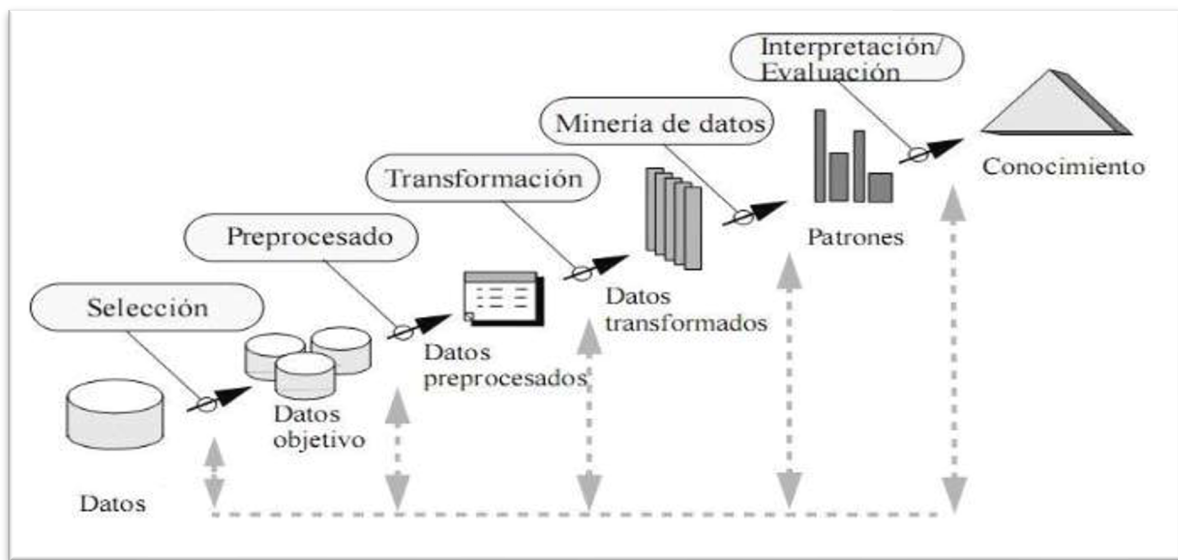


Figura 2 Etapas de KDD

¹ Es una empresa multinacional de software, destacada como una de las principales en su campo.

Al sumergirnos en la elección de la metodología para este proyecto de minería de datos, es esencial comprender las características distintivas que hacen de KDD (Knowledge Discovery in Databases) una opción relevante y efectiva. A continuación, se explorarán detalladamente las características clave que definen la metodología KDD:

- **Enfoque estructurado:** La metodología KDD divide el proceso en etapas lógicas y secuenciales, que incluyen la comprensión del negocio, la comprensión de los datos, la preparación de los datos, el modelado, la evaluación y el despliegue. Este enfoque estructurado garantiza que no se pasen por alto pasos críticos y que se siga un proceso sistemático.
- **Enfoque centrado en el negocio:** KDD pone un fuerte énfasis en comprender las necesidades y los objetivos del negocio. Esto es esencial para asegurarse de que las recomendaciones generadas por el sistema sean relevantes y valiosas para los usuarios y cumplan con los objetivos comerciales.
- **Selección y preparación de datos:** KDD subraya la importancia de la selección de datos relevantes y la preparación adecuada de los mismos. Esto es crucial en un sistema de recomendación, ya que la calidad de los datos y la representación de los usuarios y elementos tienen un impacto significativo en la calidad de las recomendaciones.
- **Modelado y evaluación:** KDD proporciona una estructura sólida para el modelado de datos y la evaluación de modelos. En un sistema de recomendación, es esencial crear modelos efectivos que puedan generar recomendaciones precisas y útiles, y KDD brinda un marco sólido para abordar esta parte crítica.
- **Enfoque iterativo:** La metodología KDD reconoce que el proceso de descubrimiento de conocimientos es iterativo. Esto es especialmente relevante en sistemas de recomendación, ya que los modelos y las recomendaciones pueden evolucionar con el tiempo a medida que se recopilan más datos y se obtienen más retroalimentaciones de los usuarios.
- **Énfasis en la interpretación:** KDD no se trata solo de construir modelos, sino de comprender y comunicar los resultados de manera efectiva. Esto es esencial en sistemas de recomendación para explicar por qué se hacen ciertas recomendaciones y ganar la confianza de los usuarios.

Se ha establecido que la presente investigación se centrará específicamente en las etapas de Selección de Datos, Minería de Datos, y Evaluación y Validación. Estas fases representan un enfoque dirigido a la identificación, análisis y evaluación de patrones de interés en la base de datos experimental.

1.6 Herramientas y Tecnologías utilizadas

Estas herramientas CASE modelan la información de negocios cuando ésta se transfiere entre distintas entidades organizativas en el seno de una compañía. El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones, y ayuda a comprender mejor la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas no satisfacen las necesidades de la organización (Asensio, 2020).

1.6.1 Herramientas de Desarrollo

Visual Studio Code v1.82

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Es software libre y multiplataforma, está disponible para Windows, GNU/Linux y macOS. VS Code tiene una buena integración con Git, cuenta con soporte para depuración de código, y dispone de un sinnúmero de extensiones, que básicamente te da la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Características destacadas:

- **Multiplataforma:** Es una característica importante en cualquier aplicación y más si trata de desarrollo. Visual Studio Code está disponible para Windows, GNU/Linux y macOS.
- **IntelliSense:** Esta característica está relacionada con la edición de código, autocompletado y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código. Como su nombre lo indica, proporciona sugerencias de código y terminaciones inteligentes en base a los tipos de variables, funciones, etc. Con la ayuda de extensiones se puede personalizar y conseguir un IntelliSense más completo para cualquier lenguaje.

- **Depuración:** Visual Studio Code incluye la función de depuración que ayuda a detectar errores en el código. De esta manera, nos evitamos tener que revisar línea por línea a puro ojo humano para encontrar errores. VS Code también es capaz de detectar pequeños errores de forma automática antes de ejecutar el código o la depuración como tal.
- **Uso del control de versiones:** Visual Studio Code tiene compatibilidad con Git, por lo que puedes revisar diferencias o lo que conocemos con git diff, organizar archivos, realizar commits desde el editor, y hacer push y pull desde cualquier servicio de gestión de código fuente (SMC). Los demás SMC están disponible por medio de extensiones.
- **Extensiones:** Hasta ahora, he mencionado varias veces el término extensiones porque es uno de los puntos fuertes. Visual Studio Code es un editor potente y en gran parte por las extensiones. Las extensiones nos permiten personalizar y agregar funcionalidad adicional de forma modular y aislada. Por ejemplo, para programar en diferentes lenguajes, agregar nuevos temas al editor, y conectar con otros servicios. Realmente las extensiones nos permiten tener una mejor experiencia, y lo más importante, no afectan en el rendimiento del editor, ya que se ejecutan en procesos independientes (Flores, 2022).

Visual Paradigm v.17.1

Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades: Enterprise, Professional, Community, Standard, Modeler y Personal. Existe una alternativa libre y gratuita de este software, la versión Visual Paradigm UML 6.4 Community Edition (Community Edition, ya que existe la Enterprise, Professional, etc.). Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de software de forma fiable a través de la utilización de un enfoque Orientado a Objetos (PARADIGM, 2018).

PgAdmin4

PgAdmin es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, seleccionado por el equipo de desarrollo, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows, por lo que la dependencia de plataforma no sería inconveniente para el desarrollo de la solución propuesta. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración (Ibarra Aliaga, Miani Flores, 2019).

1.6.2 Lenguajes de Desarrollo

Python v.3.11.5

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

Los beneficios de Python incluyen los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Los desarrolladores pueden utilizar Python fácilmente con otros lenguajes de programación conocidos, como Java, C y C++.

- La comunidad activa de Python incluye millones de desarrolladores alrededor del mundo que prestan su apoyo. Si se presenta un problema, puede obtener soporte rápido de la comunidad.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix (Amazon, 2019).

HTML 5

HTML es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto definiéndolo de forma sencilla, "HTML es lo que se utiliza para crear todas las páginas web de Internet". Más concretamente, HTML es el lenguaje con el que se "escriben" la mayoría de páginas web. Los diseñadores utilizan el lenguaje HTML para crear sus páginas web, los programas que utilizan los diseñadores generan páginas escritas en HTML y los navegadores que utilizamos los usuarios muestran las páginas web después de leer su contenido HTML. Aunque HTML es un lenguaje que utilizan los ordenadores y los programas de diseño, es muy fácil de aprender y escribir por parte de las personas. En realidad, HTML son las siglas de HyperText Markup Language y más adelante se verá el significado de cada una de estas palabras (JIMÉNEZ ALVARADO, 2018).

CSS 3

CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc. Una vez creados los contenidos, se utiliza el lenguaje

CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc (Eguiluz Pérez, 2008).

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java (Eguiluz, 2018).

Bootstrap 5

El framework Bootstrap 5 es una de las versiones recientemente lanzadas del Framework Bootstrap para el desarrollo web. Esta es una de las librerías más conocidas, pues se pueden construir aplicaciones web adaptables para móvil con el CDN de open source jsDelivr y una página con una plantilla de inicio.

En términos de la versión 5, además de algunas mejoras sobre muchas herramientas, se tomaron decisiones relacionadas al uso y a la aplicación del framework. Por ejemplo, esta versión no es compatible con Internet Explorer ni con JQuery. Asimismo, se integraron con el generador de sitios web estáticos Hugo, dejando de lado a Jekyll.

Por otro lado, se agregaron variables CSS, perfeccionaron el Grid CSS y se permitió añadir una nueva API de interfaz con Sass, donde puedes incluir tus propias herramientas (KeepCoding, 2022).

Django v.4.2.5

El framework Django es un software que puede utilizar para desarrollar aplicaciones web de forma rápida y eficiente. La mayoría de las aplicaciones web tienen varias funciones comunes, como la autenticación, la recuperación de información de una base de datos y la administración de cookies. Los desarrolladores tienen que codificar una funcionalidad similar en cada aplicación web que escriban.

Django facilita su trabajo al agrupar las diferentes funciones en una gran colección de módulos reutilizables, llamada marco de aplicación web. Los desarrolladores utilizan el marco web de Django para organizar y escribir su código de manera más eficiente y reducir significativamente el tiempo de desarrollo web (Amazon, 2021).

1.6.3 Sistemas Gestores de Bases de Datos

Un sistema gestor de base de datos (SGBD) o Database Management System (DBMS) es un conjunto de programas invisibles para el usuario final con el que se administra y gestiona la información que incluye una base de datos. Los gestores de datos o gestores de base de datos permiten administrar todo acceso a la base de datos, pues tienen el objetivo de servir de interfaz entre esta, el usuario y las aplicaciones. Entre sus funciones se encuentran la de permitir a los usuarios de negocio almacenar la información, modificar datos y acceder a los activos de conocimiento de la organización. Asimismo, el gestor de base de datos también se ocupa de realizar consultas y hacer análisis para generar informes. Además, los sistemas de gestión de base de datos pueden entenderse como una colección de datos interrelacionados, estructurados y organizados en el ecosistema formado por dicho conjunto de programas que acceden a ellos y facilitan su gestión (Darias Pérez, 2021).

PostgreSQL 14

Es un sistema de base de datos relacional de alta disponibilidad. Es capaz de funcionar de manera estable en el servidor y, por lo tanto, resulta robusto una de las principales características que buscan las empresas. Además, es consistente y tolerante a fallos. Es compatible con el modelo relacional que ya asegura siempre su integridad referencial.

Algunas de sus principales características son:

- **Alta concurrencia:** Es capaz de atender a muchos clientes al mismo tiempo y entregar la misma información de sus tablas, sin bloqueos.
- **Soporte para múltiples tipos de datos de manera nativa:** Ofrece los tipos de datos habituales en los sistemas gestores, pero además muchos otros que no están disponibles en otros competidores, como direcciones IP, direcciones MAC, Arrays, números decimales con precisión configurable, figuras geométricas, etc.

- **Soporte a triggers:** Permite definir eventos y generar acciones cuando estos se disparan.
- **Trabajo con vistas:** Esto quiere decir que pueden consultar los datos de manera diferente al modo en el que se almacenan.
- **Objeto-relacional:** Otra de sus principales características, que permite trabajar con sus datos como si fueran objetos y ofrece mecanismos de la orientación a objetos, como herencia de tablas.
- **Soporte para bases de datos distribuidas:** Donde el trabajo con transacciones asegura que estas tendrán éxito cuando han podido realizarse en todos los sistemas involucrados.
- **Soporte para gran cantidad de lenguajes:** PostgreSQL es capaz de trabajar con funciones internas, que se ejecutan en el servidor, escritas en diversos lenguajes como C, C++, Java, PHP, Python o Ruby. Además, ofrece interfaces para ODBC y JDBC, así como interfaces de programación para infinidad de lenguajes de programación (Darias Pérez, 2021).

Conclusiones parciales

En este capítulo, se ha establecido una sólida base teórica que sustenta la investigación en curso. Se han definido conceptos clave relacionados con el tema, se han explorado tecnologías y tendencias actuales, y se ha seleccionado una metodología de desarrollo adecuada, por lo tanto se arriba a las siguientes conclusiones:

1. La investigación de los principales conceptos relacionados a la búsqueda de la solución del trabajo de investigación y el estudio de los aplicaciones similares permitió identificar las funcionalidades que formarían parte de la aplicación a desarrollar.
2. Se ha proporcionado una visión general de las tecnologías actuales relevantes para el desarrollo de chatbots, incluyendo BiblioCommons, LibraryThing for Libraries y GoodReads.
3. Se definió AUP versión UCI como metodología para guiar el proceso de desarrollo, pues es una perfecta metodología para equipos d trabajos pequeños de hasta una persona.
4. Se han identificado varias herramientas de desarrollo esenciales, como Visual Studio Code, Visual Paradigm, PgAdmin 4 y lenguajes de programación como Python, HTML, CSS,

CAPÍTULO 1 FUNDAMENTACIÓN TEÓRICA

JavaScript, y frameworks como Bootstrap y Django. Estas herramientas proporcionan los medios para construir y diseñar la solución de manera efectiva.

Capítulo 2: Análisis y Diseño de la Propuesta de Solución

En el presente capítulo se muestra el modelo de dominio para representar la estructura de la solución, así como la relación entre las diferentes clases con el objetivo de vincular los conceptos asociados a la investigación. Se realiza el levantamiento y especificación de requisitos, tanto funcionales como no funcionales necesarios a cumplir para la elaboración de la solución. Se formula la propuesta de solución, así como la arquitectura y los patrones empleados en el desarrollo de la aplicación.

2.1 Descripción de la propuesta de solución

Utilizando la información recopilada en el capítulo anterior se propone el desarrollo de un módulo para el portal web de la biblioteca de la UCI que personalice la experiencia de los estudiantes y profesores al momento de buscar nuevos materiales para su lectura.

El módulo habilitará a los usuarios para efectuar consultas al portal web de la biblioteca, con la finalidad de localizar y recuperar literaturas acorde sus intereses. En el marco de dicho módulo, las obras pertinentes se presentarán de manera segmentada y clasificada en diversas sesiones temáticas. Esta disposición permitirá a los usuarios efectuar búsquedas acotadas y específicas de conformidad con sus preferencias y los temas de lectura que les susciten interés. Incluye además una funcionalidad de búsqueda avanzada que ofrezca a los usuarios la capacidad de aplicar filtros específicos para refinar sus conjuntos de resultados de búsqueda. Adicionalmente, se requiere la implementación de un mecanismo de recomendación que presente a los usuarios libros que aún no han tenido interacción con ellos, con el objetivo de fomentar la exploración y el descubrimiento de nuevos materiales bibliográficos. En paralelo, se incluye un componente en la interfaz de usuario que exhiba un conjunto de libros destacados, basados en la popularidad dentro de la comunidad de usuarios del portal. Estas funcionalidades contribuirán de manera significativa a la mejora de la experiencia del usuario, al proporcionar herramientas avanzadas para la búsqueda y presentar recomendaciones personalizadas y populares.

2.2 Usuarios relacionados con el sistema

Tabla 3 Usuarios del sistema

Usuario	Justificación
---------	---------------

Anónimo	Todos los usuarios deben autenticarse para acceder a las funcionalidades implementadas por el módulo. Este usuario puede navegar por el sistema sin obtener ningún privilegio.
Usuario Autenticado	Persona que tendrá acceso al sistema como usuario estándar, teniendo la posibilidad de recibir recomendaciones.
Administrador	Los administradores disponen de posibilidades ilimitadas para ejecutar todas las funciones administrativas del portal y el módulo.

2.3 Modelo de Dominio

Un modelo de dominio es una representación de conceptos en dominio del problema. Un modelo conceptual es un artefacto de la disciplina de análisis, construido con las reglas de UML durante la fase de concepción, presentado como uno o más diagramas de clases. Se pueden utilizar para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de software o de otro tipo. Los objetos del dominio o clases pueden obtenerse a partir de una especificación de requisitos (Pressman, 2014).

A continuación, en la figura 3 se representa el modelo conceptual.

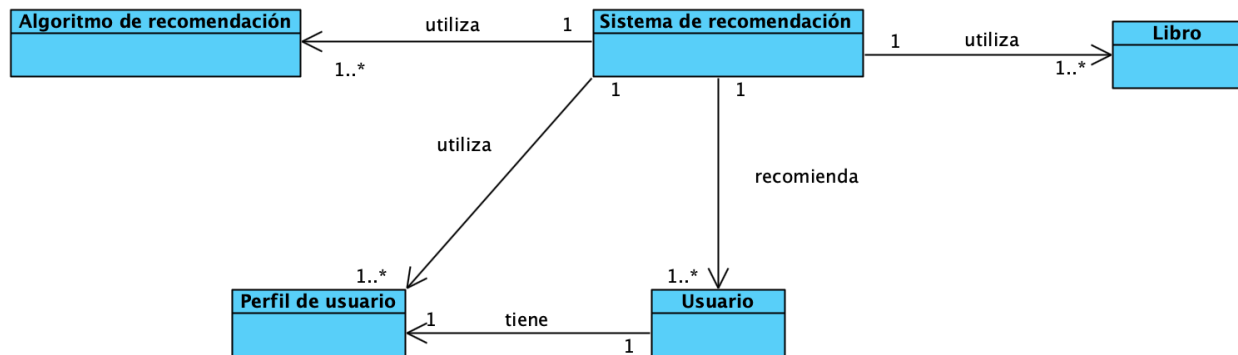


Figura 3 Modelo de dominio

Descripción de las clases:

- **Algoritmo de recomendación:** Un conjunto de reglas y procesos que analizan datos para proporcionar sugerencias personalizadas basadas en las preferencias y comportamientos del usuario.
- **Sistema de recomendación:** Una plataforma o aplicación que utiliza algoritmos para sugerir contenido, productos o servicios específicos a los usuarios, mejorando así su experiencia y satisfacción.
- **Libro:** Una obra escrita, impresa o digital, que presenta información, narrativa o conocimiento.
- **Perfil de usuario:** Una representación detallada de las características, preferencias y comportamientos de un usuario en una plataforma, utilizado para personalizar y mejorar las recomendaciones.
- **Usuario:** Una persona que utiliza un sistema o plataforma, y cuyas interacciones y preferencias son consideradas por el sistema de recomendación para proporcionar sugerencias adaptadas a sus gustos individuales.

2.4 Fundamentación del tipo sistema recomendador seleccionado

Luego de realizado un estudio bibliográfico respecto a los sistemas recomendadores, se ha decidido darle solución al problema planteado 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.

Fases 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.

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:

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

Ecuación 4 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 solo 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 estos 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 Datos⁶ 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, 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 de 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 descendientemente 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 libros 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 que predice el enfoque para U , un usuario activo U y un producto objetivo i .

A continuación se muestra la fórmula matemática de esta técnica (Moya, 2013):

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

Ecuación 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 .

$\text{sim}(u;n)$: 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 4 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 sería 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 ítems recomendados (Z_u) a los usuarios del filtrado colaborativo.

Figura 5 Recomendación de los n ítems de todos los posibles

2.5 Especificación de requisitos

2.5.1 Requisitos Funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este 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 (Sommerville, 2016).

Tabla 4 *Requisitos funcionales*

No.	Requisitos	Descripción	Complejidad	Prioridad
RF 1	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
RF 2	Establecer usuarios semejantes	El sistema establece los usuarios más similares al usuario activo a partir de su respectivo índice de correlación.	Media	Alta
RF 3	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
RF 4	Calcular Predicción de evaluación por usuario.	El sistema obtiene la predicción de evaluación que le daría el usuario activo a un libro n que ha evaluado teniendo en cuenta los usuarios similares y la evaluación que estos le dieron a n .	Alta	Alta
RF 5	Generar las recomendaciones basadas en ítem	El sistema genera una lista con un máximo de 10 recomendaciones basadas en ítem.	Media	Media
RF 6	Mostrar Recomendaciones basadas en ítem	El sistema muestra una lista de recomendaciones basadas en ítem.	Media	Alta
RF 7	Generar Recomendaciones basadas en usuario	El sistema genera una lista con un máximo de 10 recomendaciones basadas en usuario.	Media	Media

RF 8	Mostrar Recomendaciones basadas en usuario	El sistema muestra una lista de recomendaciones basadas en usuario.	Media	Alta
RF 9	Mostrar libros relacionados	Cuando el usuario visita un libro aparece una pestaña “Libros relacionados” la cual muestra una lista de libros relacionados al visitado.	Media	Media
RF 10	Mostrar libros recomendados	En el módulo se muestra una lista de libros recomendados por el sistema donde unen las recomendaciones basadas por ítem y por usuario.	Media	Alta

2.5.2 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 (Sommerville, 2016).

Tabla 5 Requisitos no funcionales

No.	Requisitos	Descripción
RnF 1	Usabilidad	- El sistema podrá ser usado por personas con conocimiento básico de tecnología. - Tipo de aplicación informática: Aplicación WEB
RnF 2	Rendimiento	- El sistema web deberá reaccionar a una velocidad de 5 Segundos a las solicitudes que el usuario necesite. - Navegar sin retrasos, el portal no podrá demorarse demasiado en realizar las peticiones de los usuarios, debe tener la capacidad de responder de forma oportuna a cada una de las

		solicitudes del usuario sin generar contratiempos, pues esto determina el tiempo de respuesta.
RnF 3	Software	- El sistema web será compatible con todos los navegadores web. - Sistema operativo Windows, Linux o MacOS
RnF 4	Hardware	- Periféricos: Mouse y Teclado. - Tarjeta de Red. - 521 MB de RAM. - Procesador Pentium 4 (o similar). - 30 GB de espacio en disco. - Conexión a Intranet
RnF 5	Apariencia o interfaz externa	- Usará protocolos de comunicación de paquetes básicos como TCP / IP y HTTPS. - Tendrá el logo de la Universidad de las Ciencias Informáticas. - Tendrá un color azul predominante

2.6 Historias de Usuario

Las historias de usuario son descripciones muy cortas y esquemáticas, que resumen la necesidad concreta de un usuario al utilizar un producto o servicio, así como la solución que la satisface. Como muchas otras herramientas ágiles, las historias de usuario surgieron como una respuesta orientada al sector de desarrollo de software. Su función principal es identificar problemas percibidos, proponer soluciones y estimar el esfuerzo que requieren implementar las ideas propuestas (Molina, 2013). El resto de las historias de usuario se encuentran en el anexo 1.

Tabla 6 HU Mostrar Recomendaciones basadas en ítem

Historia de Usuario	
Número: 6	Nombre: Mostrar Recomendaciones basadas en ítem
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Estimación: 0.2	
Descripción: Objetivo: Permite mostrar una lista de recomendaciones basada en los libros visitados por el usuario. 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. 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.	

2.7 Arquitectura del Sistema

Según IEEE (Institute of Electrical and Electronics Engineers) “La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución [...]”

La arquitectura del software de un programa o sistema de cómputo es la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos. Es una representación que permite: analizar la efectividad del diseño para cumplir los requerimientos establecidos, considerar alternativas arquitectónicas en una etapa en la que hacer cambios al diseño todavía es relativamente fácil y reducir los riesgos asociados con la construcción del software (Pressman, 2014).

2.7.1 Patrón Arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos, ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor (Pressman, 2014).

Arquitectura Modelo Vista Plantilla (MVT)

Diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales están dadas por el hecho de que, el modelo, las vistas y las plantillas se tratan como entidades separadas. Esto hace que cualquier cambio producido en el modelo se refleje de forma automática en cada una de las vistas (Romero & González, 2012).

Sin embargo, en el momento en que aparecen las bases de datos y los modelos, este proceso se extiende. Ahora se recibirá la petición, se pasará a la vista, en la vista recuperaremos los datos del modelo correspondiente, y finalmente la renderizaremos la Plantilla, pero esta vez integrando los datos dinámicos recuperados del modelo, antes de enviar el HTML final al navegador (GitHub, 2022).

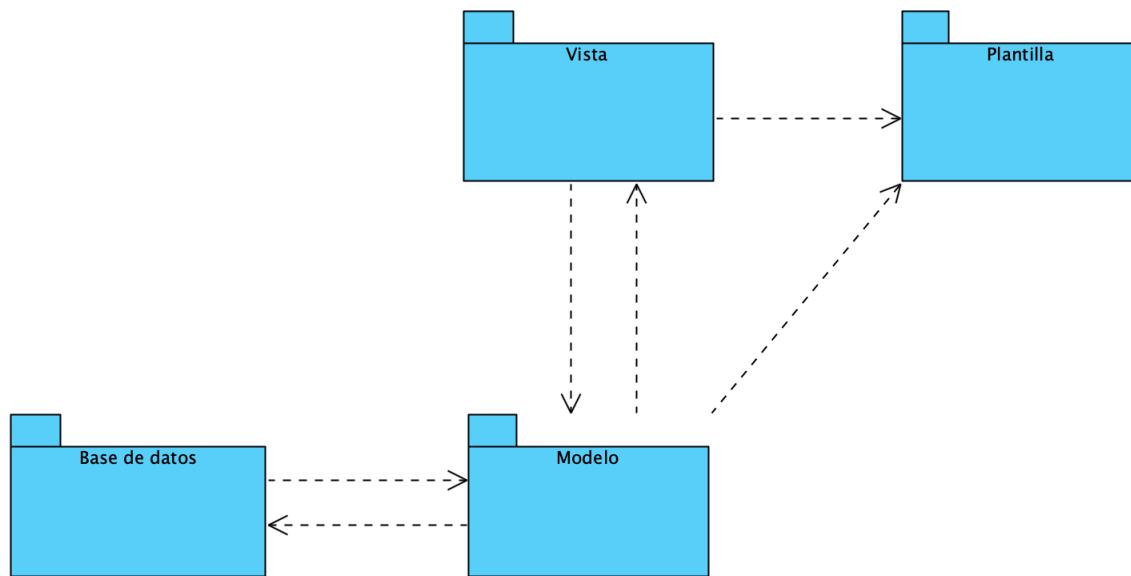


Figura 6 Representación del patrón arquitectónico

2.7.2 Patrones de Diseño

Los patrones de diseño facilitan la reutilización de diseños y arquitecturas exitosas. Un patrón de diseño sistemáticamente nombra, motiva y explica un diseño general que aborda un problema recurrente de diseño en los sistemas orientados a objetos. Describe el problema, la solución, cuándo aplicar la solución y sus consecuencias. También brinda consejos de aplicación y ejemplos. La solución es una disposición general de objetos y clases que resuelven el problema. La misma se adapta y se aplica para resolver el problema en un contexto particular (Labañino Urbina et al., 2020).

Patrones GRASP

Los patrones GRASP (por sus siglas en inglés, General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (Demestre & González, 2021).

Los patrones GRASP utilizados en la solución propuesta fueron:

Bajo acoplamiento: Debe haber pocas dependencias entre las clases. El propósito de este patrón es aumentar la reutilización y eliminar las dependencias entre las clases para propiciar un fácil mantenimiento y entendimiento (Larman, 2003). En una aplicación de Django, la definición de la URL y la función de vista que se llamará están débilmente acopladas.

```
from django.contrib import admin
from django.urls import path
from modulo import views
from modulo.views import RecomendacionView

urlpatterns = [
    path('admin/', admin.site.urls),
    path(' ', views.mostrar),
    path('recomendacion/<int:usuario_id>', RecomendacionView.as_view(), name='recomendacion'),
]
```

Figura 7 Fragmento de código de las url

Alta cohesión: Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Este patrón permitirá al equipo de desarrollo que la implementación de las clases encargadas de codificar la información sea fácil de comprender, reutilizar, mantener y poco susceptibles a cambios. En el sistema implementado se agruparon las clases según la funcionalidad de cada una (Larman, 2003).

Experto: Este patrón se basa en la asignación de responsabilidades, las cuales se asignan a las clases que posean la información necesaria para llevarlas a cabo. Es utilizado en la capa de abstracción del modelo de datos. Con el uso del *Object Relational Mapping* (ORM) de Django, se crean las clases que representan las entidades del modelo de datos. Asociado a cada una de estas clases, son generadas un conjunto de funcionalidades que las relacionan de forma directa con la entidad que representan. Las clases contienen toda la información necesaria de la tabla que representan en la base de datos.

```

38     def __str__(self):
39         return self.email
40
41     class Libro(models.Model):
42         titulo = models.CharField(max_length=200)
43         autor = models.CharField(max_length=100)
44         descripcion = models.TextField()
45         genero = models.CharField(max_length=50)
46         año_publicacion = models.PositiveIntegerField()
47
48         def __str__(self):
49             return self.titulo
50
51     class Lectura(models.Model):
52         usuario = models.ForeignKey(User, on_delete=models.CASCADE)
53         libro = models.ForeignKey(Libro, on_delete=models.CASCADE)
54         fecha_lectura = models.DateTimeField(auto_now_add=True)
55
56         def __str__(self):
57             return f'{self.usuario.username} leyó {self.libro.titulo}'
58

```

Figura 8 Fragmento de código del archivo models.py

2.7.3 Implementación del algoritmo siguiendo las fases de KDD

2.7.3.1 Selección de los datos

En esta etapa, se realiza la determinación de la relevancia de los datos para el análisis y la minería de conocimiento, lo que implica la identificación de las fuentes de datos adecuadas, la recopilación de conjuntos de datos pertinentes y la definición precisa de los objetivos del negocio.

La solución propuesta está basada en el desarrollo de un sistema basado en la web, es por ello que es preciso tener una base de datos que permita almacenar toda la información referente al sistema de selección basada en habilidades para la gestión del control de medicamentos. El diseño de esta base de datos es el proceso en el cual se toman los elementos conceptuales y se materializan. Estos elementos son representados mediante un modelo de datos, definido como “una herramienta para el modelado de datos que facilita la representación de entidades de una base de datos”. Este modelo está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades (Pressman, 2014).

La base de datos utilizada en esta investigación es de naturaleza experimental y consta de tres tablas cruciales: Usuario, Libro y Lectura. La tabla Libro almacena información esencial, como el título del libro, el autor, el año de publicación, la descripción y el género. La tabla Usuario, por otro lado, contiene los nombres de usuario junto con sus contraseñas correspondientes. La tabla Lectura complementa la estructura con detalles temporales, incorporando la fecha de cada lectura. Actualmente, la base de datos alberga datos relacionados con 200 libros, 100 lecturas y 50 usuarios. Esta configuración proporciona un conjunto representativo que permite evaluar la efectividad del algoritmo de

recomendación bajo diversas condiciones y escenarios de interacción. Cada elemento en la base de datos contribuye a la validación y prueba del sistema de recomendación, garantizando que las pruebas y evaluaciones se lleven a cabo en un entorno diverso y realista.

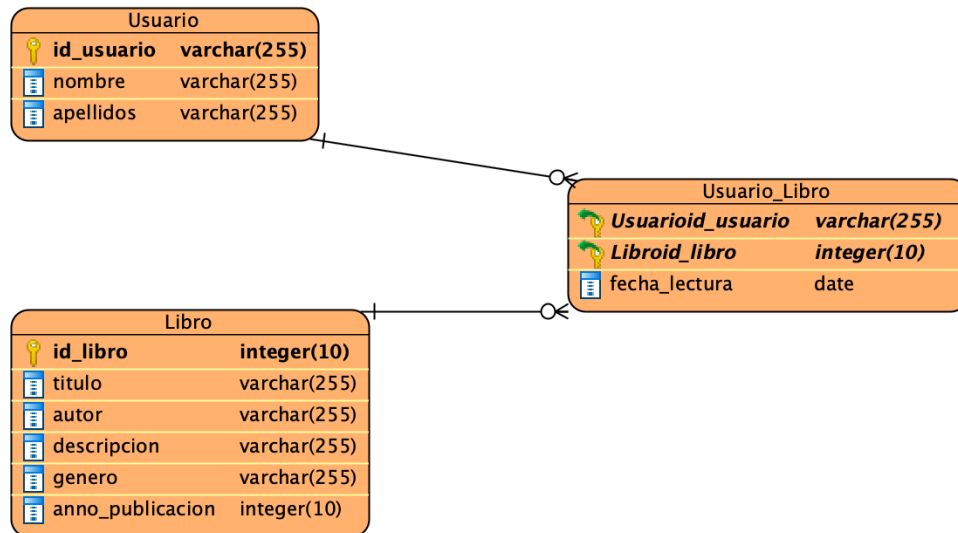


Figura 9 Modelo de datos

2.7.3.2 Minería de datos

El algoritmo de recomendación basado en filtrado colaborativo, empleando la métrica de Correlación de Pearson, desempeña un papel crucial en plataformas digitales para mejorar la experiencia del usuario. Esta métrica se utiliza para evaluar la relación lineal entre las preferencias de los usuarios, ofreciendo un enfoque efectivo para proporcionar recomendaciones personalizadas. El principal objetivo de este algoritmo es enriquecer la experiencia de los consumidores desde el inicio de su interacción con la plataforma. En lugar de depender exclusivamente de búsquedas manuales o interacciones previas, el sistema utiliza la información recopilada de diversos usuarios para prever y sugerir contenido relevante. La implementación de la métrica de Correlación de Pearson en el filtrado colaborativo capacita al algoritmo para comprender las tendencias y preferencias de los usuarios mediante la comparación de sus comportamientos con los de otros usuarios. Esto se logra evaluando la similitud en las calificaciones o interacciones pasadas entre usuarios y contenido. Un alto valor de correlación de Pearson indica una relación sólida entre las preferencias de dos usuarios, lo que se traduce en recomendaciones más precisas y personalizadas.

En el caso del método basado en ítem con media ponderada, el algoritmo realiza un análisis similar pero centrado en la relación entre los libros en lugar de los usuarios. La construcción de una matriz de interacciones binaria entre libros sienta las bases para calcular la similitud entre ítems mediante una media ponderada. Esta medida permite identificar libros similares en función de las interacciones pasadas. La selección aleatoria de libros similares garantiza una variedad en las recomendaciones, ofreciendo al usuario sugerencias diversas y relevantes. Este enfoque, basado en la similitud entre ítems, complementa el método basado en usuario, contribuyendo a una estrategia integral de recomendación que mejora la diversidad y personalización de las sugerencias.

Su funcionamiento se describe a continuación:

En el método basado en usuario, la construcción de la matriz de interacciones binaria implica crear una estructura de datos donde cada fila representa un usuario y cada columna un libro. Si un usuario ha leído un libro, se asigna un valor de 1 en la matriz; de lo contrario, se coloca un 0. Esto da como resultado una representación eficiente de las interacciones entre los usuarios y los libros en el sistema. Posteriormente, se emplea la correlación de Pearson, una medida estadística que evalúa la relación lineal entre dos conjuntos de datos, para calcular la similitud entre las preferencias de lectura de diferentes usuarios. La correlación de Pearson proporciona un coeficiente que varía entre -1 y 1, donde 1 indica una similitud perfecta, 0 significa ninguna correlación, y -1 señala una relación inversa perfecta. Una vez obtenida la matriz de similitud entre usuarios, se procede a identificar aquellos usuarios que son más similares al usuario actual. Esto se logra ordenando la matriz de similitud y seleccionando los usuarios con coeficientes más altos, excluyendo la autocorrelación (cuando el índice del usuario actual coincide con el del usuario en evaluación). Finalmente, las recomendaciones de libros se generan examinando los libros leídos por los usuarios identificados como similares al usuario actual. Se seleccionan aquellos libros que han sido apreciados por los usuarios similares pero que aún no han sido leídos por el usuario actual.

En el método basado en ítem, se establece una matriz de interacciones similar a la del método basado en usuario, pero ahora la orientación está centrada en libros en lugar de usuarios. Cada fila de la matriz representa un libro, y cada columna corresponde a un usuario. Si un libro ha sido leído por un usuario, se asigna el valor 1; de lo contrario, se coloca un 0, creando así una representación binaria de las interacciones entre libros y usuarios.

La similitud entre libros se determina a través de una media ponderada para evaluar la similitud entre los perfiles de interacciones de diferentes libros en función de la cantidad de usuarios que han leído

ambos libros. Esta medida ayuda a identificar libros que han sido consumidos por usuarios con patrones de lectura similares.

Una vez calculada la similitud entre los libros mediante la media ponderada, se procede a seleccionar libros similares para generar recomendaciones al usuario. La introducción de aleatoriedad en este proceso tiene el propósito de limitar la cantidad de recomendaciones y ofrecer una variedad en las sugerencias presentadas al usuario. Esto evita la sobreexposición a ciertos tipos de libros y enriquece la experiencia del usuario al presentar opciones diversas y relevantes.

Conclusiones Parciales

En este capítulo se realizó el análisis y diseño de la aplicación, arribando a las siguientes conclusiones:

- El levantamiento de requisitos permitió determinar las funcionalidades que debe cumplir el sistema.
- La realización del modelo conceptual permitió una mejor comprensión de los conceptos asociados al negocio.
- La implementación inicia con un filtrado colaborativo basado en la métrica de Correlación de Pearson, generando recomendaciones iniciales al considerar las preferencias de usuarios similares.
- Los artefactos generados durante el análisis y diseño de la solución contribuyeron a un mejor entendimiento del sistema para dar paso a la implementación de la solución propuesta.

Capítulo 3: Implementación y Prueba de la propuesta de solución

En el presente capítulo se presenta una descripción de la implementación del módulo a través del diagrama de despliegue y los estándares de codificación. Se define la estrategia de pruebas y se analizan los resultados de las pruebas realizadas que permiten evaluar la calidad de la propuesta de solución.

3.1 Diagrama de Despliegue

Los diagramas de despliegue se utilizan para visualizar los procesadores/ nodos/dispositivos de hardware de un sistema, los enlaces de comunicación entre ellos y la colocación de los archivos de software en ese hardware. Un diagrama de despliegue es un tipo de diagrama UML que muestra la arquitectura de ejecución de un sistema, incluyendo nodos como entornos de ejecución de hardware o software, y el middleware que los conecta.

Los diagramas de despliegue se utilizan normalmente para visualizar el hardware y el software físico de un sistema. Usándolo puedes entender cómo el sistema se desplegará físicamente en el hardware. Los diagramas de despliegue ayudan a modelar la topología de hardware de un sistema (Siriwardhana, 2020).

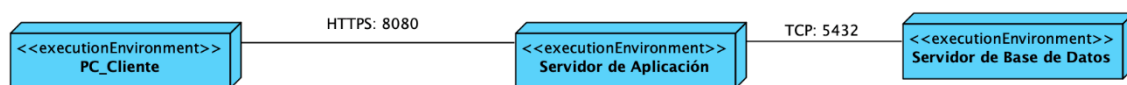


Figura 10 Diagrama de despliegue

Descripción de los Nodos

- **Servidor de Base de Datos:** Es el nodo donde se almacenan los datos del sistema mediante el sistema de Base de Datos PostgreSQL 14.

- **Servidor de Aplicaciones:** Es el nodo encargado de tener instalado el sistema al que tendrán acceso los usuarios desde las estaciones de trabajo, con el framework Django y el sistema de Base de Datos PostgreSQL 14.
- **PC Cliente:** Es el nodo que representa la estación de trabajo que permite al usuario mediante el protocolo HTTPS:8080 y el puerto 5432, utilizando los navegadores web para acceder al sistema y la impresora para otras funciones.

3.2 Estrategia de pruebas de software

Según IEEE (2004), las pruebas de software "consisten en verificar el comportamiento de un programa dinámicamente a través de un grupo finito de casos de prueba, debidamente seleccionados del, típicamente, ámbito de ejecuciones infinito, en relación al comportamiento esperado". Contrario a lo que se pueda pensar una prueba exitosa es aquella en la que se consigue que el sistema falle y por tanto, se encuentran errores, aunque estos no sean todos los que posee el producto. De igual forma se puede considerar una prueba exitosa cuando es posible asegurar que no existen más errores en el sistema, cuestión que es mucho más difícil de garantizar. Este es un aspecto del que advierte muy a menudo la teoría de pruebas, ya que no es confiable asumir que se han superado todas las pruebas posibles que pueden efectuarse a un software (Justiz-Núñez & Gómez-Suárez, 2014).

Se plantea la realización de pruebas de precisión, pruebas unitarias y pruebas de aceptación.

- Las **pruebas de precisión** son fundamentales para evaluar la eficacia y exactitud de un algoritmo en el contexto de sistemas de recomendación. Estas pruebas se diseñan para medir la capacidad del algoritmo de proporcionar recomendaciones que se alineen de manera precisa con las preferencias y comportamientos reales de los usuarios. La precisión se evalúa mediante métricas específicas, como la precisión en los primeros N elementos recomendados o la precisión global del sistema. Las pruebas de aceptación involucran la generación de artefactos, como el diseño de casos de prueba, que permiten verificar si el sistema cumple con los requisitos funcionales y las expectativas del usuario.
- Las **pruebas unitarias** se llevan a cabo mediante el archivo unittest.py del proyecto Django, asegurando la verificación a nivel de unidad de las distintas funcionalidades y componentes del software. Estas pruebas garantizan que cada unidad del sistema funcione correctamente de manera aislada, contribuyendo así a la robustez y confiabilidad del conjunto.

- Las **pruebas de aceptación** involucran la generación de artefactos, como el diseño de casos de prueba, que permiten verificar si el sistema cumple con los requisitos funcionales y las expectativas del usuario.

3.2.1 Pruebas Unitarias

Estas pruebas son pruebas de caja blanca, en la que los casos de prueba se basan en la estructura interna. El probador elige entradas para explorar rutas particulares y determina la salida apropiada. El propósito de las pruebas unitarias es examinar los componentes individuales o piezas de métodos / clases para verificar la funcionalidad, asegurando que el comportamiento sea el esperado. El alcance exacto de una “unidad” a menudo se deja a la interpretación, pero una buena regla general es que una unidad contenga la menor cantidad de código que realice una tarea independiente (por ejemplo, un solo método o clase) (Déléchamp, 2018).

Con las pruebas unitarias del software se comprueban los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

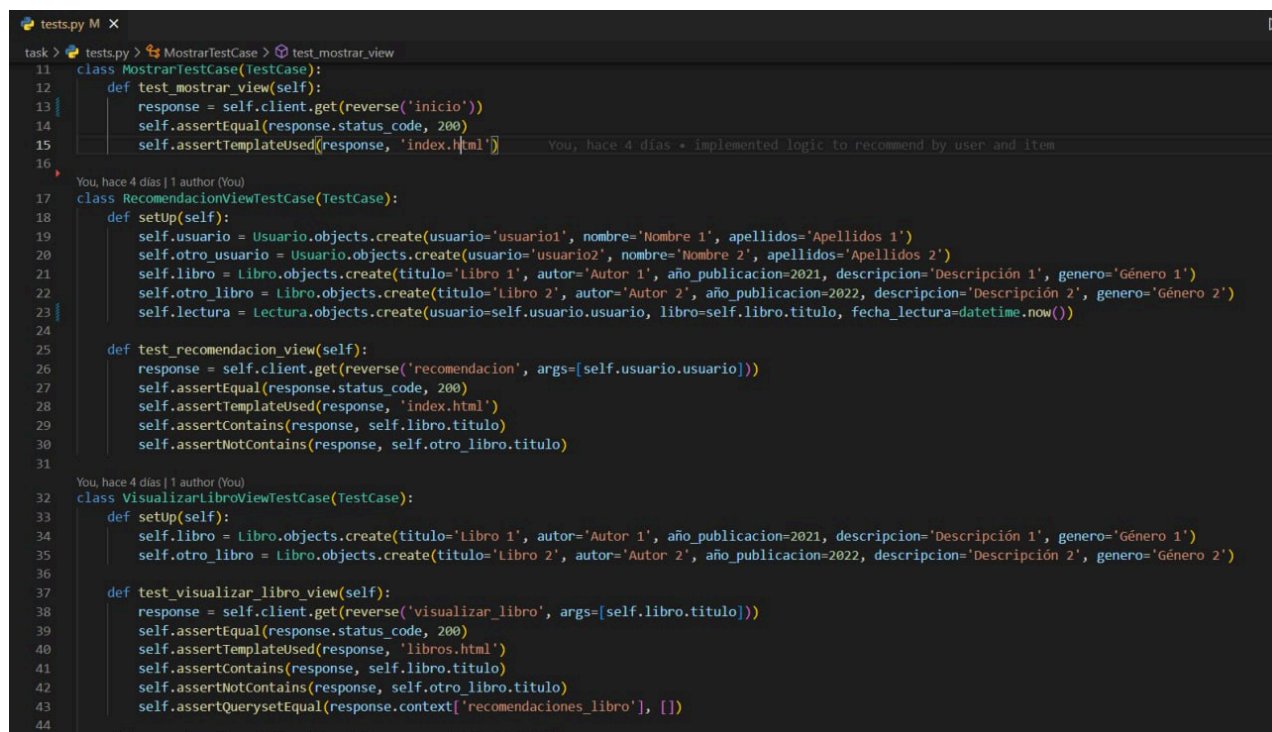
Debido a la propia filosofía de Python, es difícil encontrar tanto herramientas de prueba como depuradores. Al ser un lenguaje interpretado que no necesita compilarse ni enlazarse, se supone que el propio código (usando el intérprete) es todo lo necesario para probar y depurar un programa. Dentro de la gama de este tipo de herramientas se encuentran varias de ellas como: PyUnit, Py.Test, TestOOB, NOSE, TRIAL y QUNITTEST, de las cuales la más factible fue PyUnit.

PyUnit es la herramienta oficial para hacer pruebas unitarias en Python. Se incluye en la librería estándar (en el módulo unittest) y ha sido creado por los desarrolladores de JUnit para facilitar la creación y gestión de tests de prueba en módulos Python. Es soportado perfectamente por el IDE Eclipse con el plugin PyDev, permite separar el código de pruebas del código del propio proyecto para mejorar la comprensión, refactorización y la facilidad para hacer cambios en el programa. Por otra parte, se pueden realizar los tests por línea de comandos.

Es capaz de realizar son pruebas de igualdad, excepciones, fallo, pruebas de cordura (comprobar que no se producen situaciones imposibles) y pruebas de mayúsculas (relacionadas con las cadenas de

caracteres). Debido a todo esto, y a que se trata de la herramienta estándar de Python, tiene mucha difusión y existe mucha documentación sobre esta librería en la red.

Para poder hacer un caso de prueba propio en PyUnit es necesario hacer una subclase de "TestCase", que es un objeto que puede correr un único método de prueba. Para realizar una prueba de algo en Python se usa el comando "assert", si este comando falla cuando se ejecuta el caso de prueba PyUnit dará el caso por fallido, de lo contrario el caso el resultado es correcto. Para cada método se le hicieron la cantidad de pruebas necesarias para su correcta ejecución.



```

tests.py M X
task > tests.py > MostrarTestCase > test_mostrar_view
11 class MostrarTestCase(TestCase):
12     def test_mostrar_view(self):
13         response = self.client.get(reverse('inicio'))
14         self.assertEqual(response.status_code, 200)
15         self.assertTemplateUsed(response, 'index.html')
16
17 You, hace 4 días | 1 author (You)
18 class RecomendacionViewTestCase(TestCase):
19     def setUp(self):
20         self.usuario = Usuario.objects.create(usuario='usuario1', nombre='Nombre 1', apellidos='Apellidos 1')
21         self.otro_usuario = Usuario.objects.create(usuario='usuario2', nombre='Nombre 2', apellidos='Apellidos 2')
22         self.libro = Libro.objects.create(titulo='Libro 1', autor='Autor 1', año_publicacion=2021, descripcion='Descripción 1', genero='Género 1')
23         self.otro_libro = Libro.objects.create(titulo='Libro 2', autor='Autor 2', año_publicacion=2022, descripcion='Descripción 2', genero='Género 2')
24         self.lectura = Lectura.objects.create(usuario=self.usuario.usuario, libro=self.libro.titulo, fecha_lectura=datetime.now())
25
26     def test_recomendacion_view(self):
27         response = self.client.get(reverse('recomendacion', args=[self.usuario.usuario]))
28         self.assertEqual(response.status_code, 200)
29         self.assertTemplateUsed(response, 'index.html')
30         self.assertContains(response, self.libro.titulo)
31         self.assertNotContains(response, self.otro_libro.titulo)
32
33 You, hace 4 días | 1 author (You)
34 class VisualizarLibroViewTestCase(TestCase):
35     def setUp(self):
36         self.libro = Libro.objects.create(titulo='Libro 1', autor='Autor 1', año_publicacion=2021, descripcion='Descripción 1', genero='Género 1')
37         self.otro_libro = Libro.objects.create(titulo='Libro 2', autor='Autor 2', año_publicacion=2022, descripcion='Descripción 2', genero='Género 2')
38
39     def test_visualizar_libro_view(self):
40         response = self.client.get(reverse('visualizar_libro', args=[self.libro.titulo]))
41         self.assertEqual(response.status_code, 200)
42         self.assertTemplateUsed(response, 'libros.html')
43         self.assertContains(response, self.libro.titulo)
44         self.assertNotContains(response, self.otro_libro.titulo)
45         self.assertQuerysetEqual(response.context['recomendaciones_libro'], [])

```

Figura 11 Fragmento de código del archivo unittest

Estas pruebas unitarias verifican que la vista RecomendacionView funcione correctamente y que proporcione recomendaciones basadas en ítem y usuario, incluso cuando un usuario no tiene lecturas registradas.

3.2.2 Pruebas de precisión

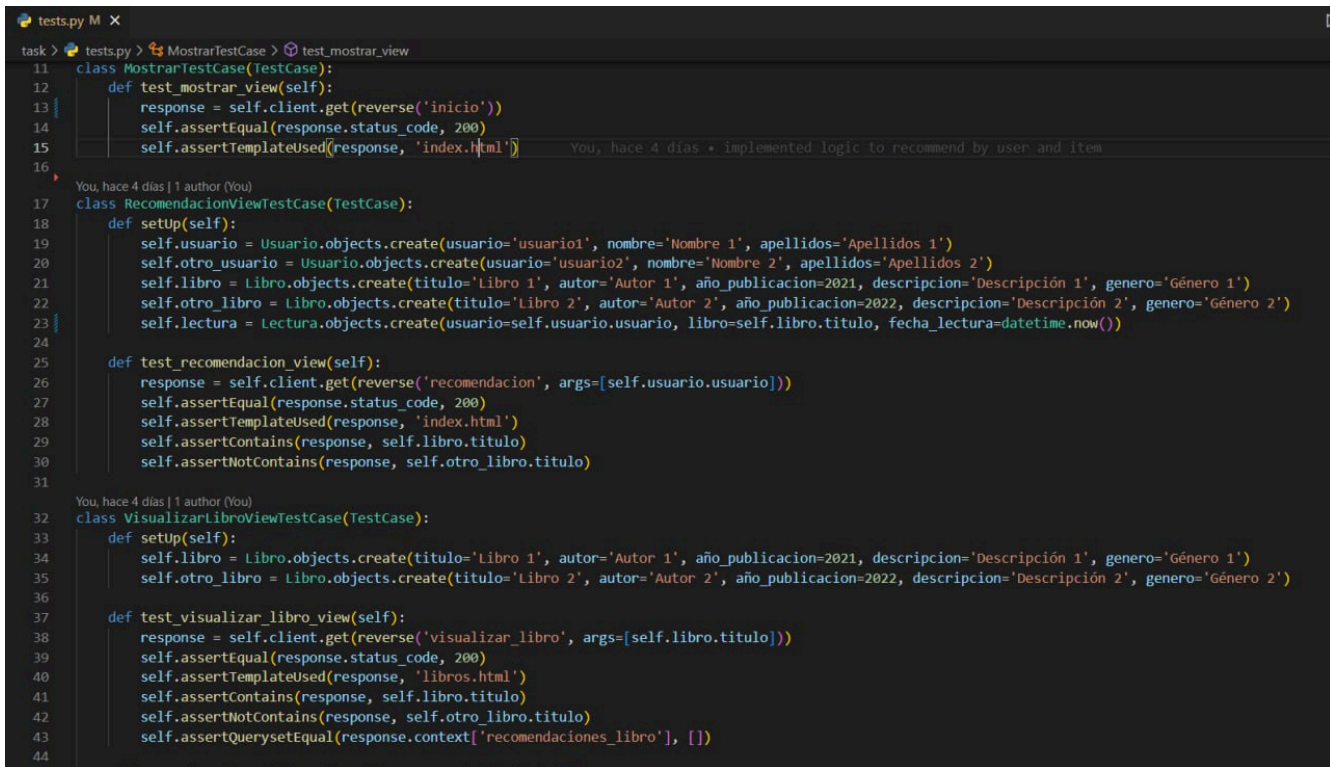
La prueba de precisión realizada tiene como objetivo evaluar el rendimiento de un algoritmo de recomendación implementado en un sistema de gestión de biblioteca. Este algoritmo recomienda libros a usuarios basándose en sus lecturas de los libros.

Pruebas automatizadas

Se deciden realizar pruebas automatizadas seleccionando 100 títulos de libros de la base de datos aleatoriamente, para ser usados como entrada de los algoritmos y obtener las diez mejores recomendaciones; estas pruebas fueron realizadas de la siguiente forma:

- En la configuración inicial de la prueba, se establecieron los modelos de datos esenciales, incluyendo los modelos de Usuario, Libro y Lectura, que representan a los usuarios, los libros y las lecturas de los usuarios en los libros en el sistema.
- Se generaron datos de prueba ficticios para simular situaciones de usuarios reales en el sistema.
- Se simulaban solicitudes de recomendaciones para usuarios específicos. En este caso, se evaluó la precisión de las recomendaciones para 'usuario1'. Se utilizó la vista 'RecomendacionView' del sistema para obtener las recomendaciones generadas por el algoritmo.
- Para calcular la precisión, se compararon las recomendaciones generadas por el algoritmo con las lecturas reales dadas por 'usuario1' en el conjunto de prueba. Si un libro recomendado estaba presente en el conjunto de prueba, se consideraba una predicción correcta. La precisión se calculó como la proporción de predicciones correctas en relación con el conjunto de prueba.
- La precisión calculada se comparó con un umbral predefinido. Si la precisión era igual o superior al umbral, se consideró que el algoritmo generaba recomendaciones precisas para el usuario evaluado.

A continuación se presenta un fragmento del código del archivo prueba_precisión.py que contiene la prueba de precisión:



```

11 class MostrarTestCase(TestCase):
12     def test_mostrar_view(self):
13         response = self.client.get(reverse('inicio'))
14         self.assertEqual(response.status_code, 200)
15         self.assertTemplateUsed(response, 'index.html')
16
17 You, hace 4 días | 1 author (You)
18 class RecomendacionViewTestCase(TestCase):
19     def setUp(self):
20         self.usuario = Usuario.objects.create(usuario='usuario1', nombre='Nombre 1', apellidos='Apellidos 1')
21         self.otro_usuario = Usuario.objects.create(usuario='usuario2', nombre='Nombre 2', apellidos='Apellidos 2')
22         self.libro = Libro.objects.create(titulo='Libro 1', autor='Autor 1', año_publicacion=2021, descripcion='Descripción 1', genero='Género 1')
23         self.otro_libro = Libro.objects.create(titulo='Libro 2', autor='Autor 2', año_publicacion=2022, descripcion='Descripción 2', genero='Género 2')
24         self.lectura = Lectura.objects.create(usuario=self.usuario.usuario, libro=self.libro.titulo, fecha_lectura=datetime.now())
25
26     def test_recomendacion_view(self):
27         response = self.client.get(reverse('recomendacion', args=[self.usuario.usuario]))
28         self.assertEqual(response.status_code, 200)
29         self.assertTemplateUsed(response, 'index.html')
30         self.assertContains(response, self.libro.titulo)
31         self.assertNotContains(response, self.otro_libro.titulo)
32
33 You, hace 4 días | 1 author (You)
34 class VisualizarLibroViewTestCase(TestCase):
35     def setUp(self):
36         self.libro = Libro.objects.create(titulo='Libro 1', autor='Autor 1', año_publicacion=2021, descripcion='Descripción 1', genero='Género 1')
37         self.otro_libro = Libro.objects.create(titulo='Libro 2', autor='Autor 2', año_publicacion=2022, descripcion='Descripción 2', genero='Género 2')
38
39     def test_visualizar_libro_view(self):
40         response = self.client.get(reverse('visualizar_libro', args=[self.libro.titulo]))
41         self.assertEqual(response.status_code, 200)
42         self.assertTemplateUsed(response, 'libros.html')
43         self.assertContains(response, self.libro.titulo)
44         self.assertNotContains(response, self.otro_libro.titulo)
45         self.assertQuerysetEqual(response.context['recomendaciones_libro'], [])

```

Figura 12 Fragmento de código de prueba de precisión

Resultados de las pruebas de precisión automatizadas

Las pruebas arrojaron un valor de precisión que indicaba qué tan acertadas fueron las recomendaciones en comparación con las calificaciones reales. En particular, se encontró que el 85% de las recomendaciones ofrecidas por el sistema fueron identificadas como pertinentes y apropiadas por los usuarios. Esta tasa de precisión es un indicativo sólido de la efectividad del algoritmo subyacente utilizado para generar las recomendaciones. Esto respalda la utilidad y el potencial impacto positivo del sistema en la experiencia del usuario y su capacidad para ofrecer contenido relevante y personalizado en función de su historial de interacciones.

3.2.3 Pruebas de aceptación

Las pruebas funcionales están basadas en ejecución, revisión y retroalimentación de las funcionalidades implementadas para el software y las especificaciones definidas por el usuario. Con estas pruebas se busca evaluar el sistema mediante modelos de pruebas. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Por

CAPÍTULO 3 IMPLEMENTACIÓN Y PRUEBA

ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Cuando se crea un software personalizado para un cliente, se realizan una serie de pruebas de aceptación para que el cliente pueda validar todos los requisitos. Conducido por el usuario final en lugar de por los ingenieros de software, una prueba de aceptación puede ir desde una “prueba de manejo” informal hasta una serie de pruebas planificadas y ejecutadas sistemáticamente. De hecho, las pruebas de aceptación pueden llevarse a cabo durante un período de semanas o meses, descubriendo así errores acumulativos que podrían degradar el sistema con el tiempo.

Las pruebas de aceptación son importantes dado que significan la satisfacción del cliente con el producto desarrollado, el final de una iteración y el comienzo de la siguiente. Se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra (Pressman, 2014). En la realización de estas pruebas se definieron los siguientes elementos:

CU: Número de la HU a la cual pertenece.

Nombre: Junto al código conforma el identificador del caso de prueba.

Descripción: Acción que debe realizar el sistema.

Condiciones de ejecución: Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.

Entrada/Pasos de Ejecución: Incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.

Resultados Esperados: Descripción de la respuesta del sistema ante el caso de prueba.

Evaluación de la prueba: Clasificación de la prueba en satisfactoria o insatisfactoria.

Tabla 7 Caso de prueba de aceptación #1

Caso de Prueba de Aceptación	
Código: HU-P1	Historia de Usuario: HU-8
Nombre: Mostrar recomendaciones basadas en usuario.	
Descripción: Se muestran las recomendaciones basadas en el usuario.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema y debe haber leído al menos un libro.	

Entradas/Pasos de ejecución: El usuario accede a la página inicial y se muestran las recomendaciones de libros basadas en algún usuario.
Resultados esperados: Se muestran las recomendaciones basadas en el usuario.
Evaluación de la prueba: Satisfactoria

Tabla 8 Caso de prueba de aceptación #2

Caso de Prueba de Aceptación	
Código: HU9-P2	Historia de Usuario: HU-9
Nombre: Mostrar libros relacionados.	
Descripción: Se muestran las recomendaciones basadas en el libro que el usuario visita.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: El usuario visita un libro, y en la parte inferior se muestra una sección con los libros similares al mismo.	
Resultados esperados: Se muestran las recomendaciones basadas en el libro visitado.	
Evaluación de la prueba: Satisfactoria	

Tabla 9 Caso de prueba de aceptación #3

Caso de Prueba de Aceptación	
Código: HU10-P3	Historia de Usuario: HU-10
Nombre: Mostrar libros recomendados.	
Descripción: Se muestra un listado de libros.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Entradas/Pasos de ejecución: En la página inicial se muestran recomendaciones de libros basadas en ítem y en usuarios.	
Resultados esperados: se muestra un listado de recomendaciones de libros.	
Evaluación de la prueba: Satisfactoria	

Resultados de las pruebas de aceptación

Se realizaron tres iteraciones donde se encontraron 7 no conformidades entre la primera y la segunda iteración, las que fueron mitigadas en la tercera iteración para lograr la calidad y el correcto funcionamiento del sistema.

Se detectaron errores ortográficos fundamentalmente en las no conformidades no significativas, en el caso de las significativas, se detectaron errores de validación. Las pruebas se realizaron de forma iterativa e incremental, comprobando en cada iteración que hubiesen sido corregidos los errores detectados en la iteración anterior, lo que contribuyó a mejorar la calidad y funcionalidad del software.

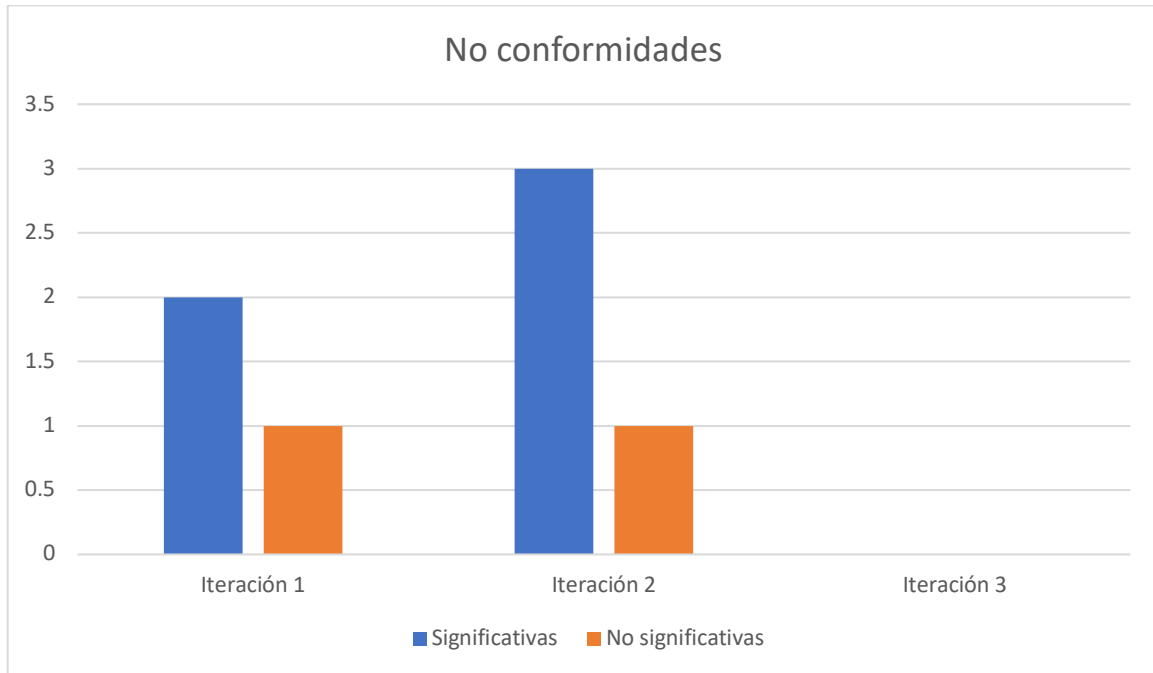


Figura 13 No conformidades detectadas

Conclusiones parciales

Como resultado del desarrollo de este capítulo, se generó el diagrama de despliegue, así como la descripción de los casos de prueba con el objetivo de que los mismos sean usados para verificar el correcto funcionamiento del módulo. 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 biblioteca, el cual reducirá el tiempo de búsqueda de libros a los usuarios según su interacción con el sistema.

Conclusiones Generales

A partir de la investigación realizada se llegan a las siguientes conclusiones:

- 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 módulo recomendador 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.
- El uso del framework de aplicaciones web Django fue una gran elección para la implementación del prototipo al ser una herramienta que permite el desarrollo ágil de proyectos de pequeña a grande escala. Su principal ventaja es la facilidad de manejo incluso para desarrolladores con poca experiencia que combinado con características como su rapidez, su seguridad y escalabilidad, lo convierten en una elección sólida para la implementación del módulo.

Recomendaciones

- 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

- Amazon. (2019). *¿Qué es Python? - Explicación del lenguaje Python - AWS*. Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is/python/>
- Amazon. (2021). *¿Qué es Django? - Explicación del software Django - AWS*. Amazon Web Services, Inc. <https://aws.amazon.com/es/what-is/django/>
- Asensio, R. M. (2020, febrero 29). *Herramientas CASE. Ingeniería del software. Informática Aplicada a la gestión Pública. Universidad de Murcia*.
https://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html
- Capouya, A. (2009, marzo 24). *APUNTES DE INFORMÁTICA: Algoritmo*.
<http://informaticafrida.blogspot.com/2009/03/algoritmo.html>
- Charles, J. (2020). *LibraryThing*. LibraryThing.Com. <https://www.librarything.com>
- Cingolani, Lic. E. A. (2014a). *Evaluación de Sistemas Recomendadores de Contenidos Educativos a través de Estudios de Usuarios. S.I. : Universidad Abierta Interamericana*.
- Cingolani, Lic. E. A. (2014b). *Evaluación de Sistemas Recomendadores de Contenidos Educativos a través de Estudios de Usuarios*. Facultad de Tecnología Informática Universidad Abierta Interamericana.
- Darias Pérez, S. (2021, septiembre 9). *Gestor de Base de datos: Qué es, Funcionalidades y Ejemplos*. Intelequia. <https://intelequia.com/es/blog/post/gestor-de-base-de-datos-qu%C3%A9-es-funcionalidades-y-ejemplos>
- Déléchamp, F. (2018). *Java y Eclipse: Desarrolle una aplicación con Java y Eclipse»*.
- Demestre, D. H., & González, y. h. (2021). *Diseño de un Sistema de Gestión de Información de Recursos Humanos*. 31-40.
- Eguiluz, J. (2018). *Capítulo 1. Introducción (Introducción a JavaScript)*.
<https://uniwebidad.com/libros/javascript/capitulo-1>
- Eguiluz Pérez, J. (2008). *Introducción a CSS*.
- Escudero, H. M. (2014). *Sistema de recomendación multimedia basado en perfiles de usuarios. Cataluña: Universidad Politécnica de Catalunya*.
- Flores, F. (2022, julio 22). *Qué es Visual Studio Code y qué ventajas ofrece*. OpenWebinars.net. <https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/>
- GitHub. (2022). *19 - Patrón MVT: Modelo Vista Template · Issue #19 · Gohanckz/workshop-django*.
GitHub. <https://github.com/Gohanckz/workshop-django/issues/19>
- Ibarra Aliaga, Miani Flores, A., Marcos Agustin. (2019). *PostgreSQL*.
- JIMÉNEZ ALVARADO, Y. P. (2018). *HTML - Ensayos universitarios—7335 Palabras*. Buenas Tareas. <https://www.buenastareas.com/ensayos/Html/81314431.html>
- Justiz-Núñez, D., & Gómez-Suárez, D. (2014). *Proceso de pruebas para productos de software en un laboratorio de calidad* (Vol. 2). Ingeniería industrial.
- KeepCoding, R. (2022, septiembre 14). *¿Qué es Bootstrap 5? | KeepCoding Bootcamps*.
<https://keepcoding.io/blog/que-es-bootstrap-5/>
- Labañino Urbina, S., Toledano López, O. G., & Labañino Gainza, C. (2020). *Algoritmo de búsqueda fonética para el Sistema de Gestión Integral de Aduanas*. 82-96.
- Larman, C. (2003). *UML y Patrones*.
- Moine, J. M. (2013). *Metodologías para el descubrimiento de conocimiento en bases de datos: Un estudio comparativo* [Magister en Ingeniería de Software, Universidad Nacional de La Plata].
<https://doi.org/10.35537/10915/29582>
- Molina, SILVIA gabriela R. (2013). *Metodologías ágiles enfocadas al modelado de requerimientos. Informes Científicos Técnicos-UNPA*, 5(1), 1-29.
- Moya, R. (2013, septiembre 24). *Sistemas de Recomendación basados en Filtrado Colaborativo (K-*

- Vecinos)—Jarroba. <https://jarroba.com/sistemas-de-recomendacion-basados-en-filtrado-colaborativo-k-vecinos/>
- Olah, A. (2018, enero 6). Qué es Goodreads y para qué sirve. *Una Buena Lectura*. <https://unabuenalectura.com/2018/01/06/que-es-goodreads-y-para-que-sirve/>
- Orenes, N. (2018). Goodreads: La mayor red social de lectores. *Como escribir un libro - Triunfa con tu libro*. <https://triunfacontulibro.com/goodreads-la-mayor-red-social-de-lectores/>
- PARADIGM, V. (2018). *Visual Paradigm Product Overview*. https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/5963_visualparadi.html.
- Parra Valero, P. (2016). *La biblioteca integrada como modelo de convergencia educativa en alfabetización en información* [doctoralThesis]. <https://e-archivo.uc3m.es/handle/10016/23078>
- Peña, F. A. y R. C., & Ricardo Elias. (2008). *Revisión, selección e implementación de un algoritmo de recomendación de material bibliográfico utilizando tecnología J2EE*. Chile: Universidad del Bio-Bio.
- Pressman. (2014). *Ingeniería del software: Un enfoque práctico, 7ma Edición – Roger S. Pressman*. <https://www.freelibros.net/ingenieria/ingenieria-del-software-un-enfoque-practico-7ma-edicion-roger-s-pressman>
- Rensick, M. L. (2004). *Persuasive Design Through Intelligent Recommendation Systems*. S.I. : *Proc. Of the Human Factors and Ergonomics Society Annual Meeting*.
- Ricci, F., Rokach, L., & Shapira, B. (2011). *Introduction to Recommender Systems Handbook*.
- Riveros Rey, O., Romero Fajardo, J., & Herrera Cubides, J. (2017). *Implementación de la técnica de los K-Vecinos en un algoritmo recomendador para un sistema de compras utilizando NFC y Android*.
- Rodríguez Sánchez, T. (2015). *Metodología de Desarrollo para la actividad productiva de la UCI*.
- Romero, Y. F., & González, Y. D. (2012). *Patrón Modelo-Vista-Controlador*. 11, 47-57.
- Santos, M. F. (2021, septiembre 23). *BiblioCommons*. <https://oaklandlibrary.org/blogs/post/crea-tu-propia-cuenta-en-bibliocommons>
- Siriwardhana, S. (2020, octubre 26). Tutorial de Diagrama de Despliegue | ¿Qué es un Diagrama de Despliegue. *Blog de Creately*. <https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/>
- Sommerville, I. (2016). *Ingeniería del Software Séptima Edición* (Vol. 7). PEARSON ADDISON WEASLEY, SA. Madrid, 2005.
- Timarán, R. (2009). Una mirada al descubrimiento de conocimiento en bases de datos. *Ventana Informática*, 20, 39-58.
- Timarán-Pereira, S. R., Hernández-Arteaga, I., Caicedo-Zambrano, S. J., Hidalgo-Troya, A., & Alvarado-Pérez, J. C. (2016). El proceso de descubrimiento de conocimiento en bases de datos. *Descubrimiento de patrones de desempeño académico con árboles de decisión en las competencias genéricas de la formación profesional*, 63-86.
- Velez-Langs, O., & Santos, C. (2006). *Sistemas Recomendadores: Un enfoque desde los algoritmos genéticos* (Vol. 9).
- Vera, J., Villalba, K., & Ocsa Mamani, A. (2015). *Modelo de sistema de recomendación de Objetos de Aprendizaje en dispositivos móviles, caso: Creación del pensamiento computacional*.

Anexos

Anexo 1 Historias de Usuario

Tabla 10 HU Calcular índice

Historia de Usuario	
Número: 1	Nombre: Calcular índice de correlación de Pearson
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción: Objetivo: Calcula el índice de correlación de Pearson del usuario activo con los usuarios que han evaluado al menos un libro. Acciones para lograr el objetivo (precondiciones y datos): - El usuario debe estar autenticado. - El usuario debe haber evaluado al menos un libro. 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: N/A	

Tabla 11 HU Usuarios Semejantes

Historia de Usuario	
Número: 2	Nombre: Establecer usuarios semejantes
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción Objetivo: Obtener una lista de los usuarios más semejantes a usuario autenticado. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe estar autenticado.	

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.

Tabla 12 HU Ítems Semejantes

Historia de Usuario	
Número: 3	Nombre: Establecer ítems semejantes
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción Objetivo: Devuelve una lista de libros semejantes según su categoría, autor y editorial. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe visitar al menos un libro. 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: N/A	

Tabla 13 HU Predicción por usuario

Historia de Usuario	
Número: 4	Nombre: Calcular predicción de evaluación por usuario.
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción	

Objetivo: Calcular la predicción de evaluación que le daría el usuario autenticado a un libro que no ha evaluado. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe estar autenticado. 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: N/A

Tabla 14 HU Recomendaciones basadas en usuario

Historia de Usuario	
Número: 7	Nombre: Generar recomendaciones basadas en el usuario
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción Objetivo: Devuelve una lista de recomendaciones basada en los las predicciones de evaluación del usuario autenticado. Acciones para lograr el objetivo (precondiciones y datos): - El usuario debe estar autenticado. 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:	

Tabla 15 HU Recomendaciones basadas en ítem

Historia de Usuario	
Número: 6	Nombre: Generar recomendaciones basadas en ítem
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.2	
Descripción Objetivo: Devuelve una lista de recomendaciones basada en los libros visitados por el usuario. 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. 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: N/A	

Tabla 16 HU Mostrar Recomendaciones basadas en usuario

Historia de Usuario	
Número: 8	Nombre: Mostrar recomendaciones basadas en el usuario
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Estimación: 0.4	
Descripción Objetivo:	

Se muestra una lista de recomendaciones basadas en los libros visitados por el usuario. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe estar autenticado. 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_02, HU_04 y la HU_07. Se muestra un máximo de 10 recomendaciones.
Observaciones: N/A

Tabla 17 HU Mostrar Libros relacionados

Historia de Usuario	
Número: 9	Nombre: Mostrar libros relacionados
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Media	Riesgo en desarrollo: Bajo
Estimación: 0.4	
Descripción Objetivo: Muestra una lista de libros relacionados al libro visitado. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe estar autenticado. 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_03. 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.	
Observaciones: N/A	

Tabla 18 HU Mostrar Libros recomendados

Historia de Usuario	
Número: 10	Nombre: Mostrar libros recomendados por el sistema
Programador: Ezequiel Brito Navarro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Medio
Estimación: 0.4	
Descripción: Objetivo: Muestra una lista de recomendaciones basada en los libros visitados por el usuario y en los usuarios semejantes al usuario autenticado. Acciones para lograr el objetivo (precondiciones y datos): El usuario debe estar autenticado. Flujo de la acción a realizar: En la portada se muestra una lista de libros recomendados apoyado en las HU_05 y HU_07	
Observaciones: Las recomendaciones no exceden de 20 libros.	