Facultad 1

Implementación de un Sistema de Recomendación basado en Deep Learning para las plataformas del proyecto Z17.

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

Autor:
Alejandro Figueroa Rodríguez

Tutores:
MSc. Aneyty Martin García

Ing. Yosbel Falero Vento

MSc. Yadier Perdomo Cuevas

La Habana, diciembre de 2024
Año 66 de la Revolución

DECLARACIÓN DE AUTORÍA

El autor Alejandro Figueroa Rodríguez del trabajo de diploma con título ***"Implementación de un Sistema de Recomendación Basado en Deep Learning para las plataformas del proyecto Z17"***, 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 firma(n) la presente a los <día> días del mes de <mes> del año <año>.

Alejandro Figueroa Rodríguez	**MSc. Aneyty Martin García**
_____	_____
Firma del Autor	Firma del Autor
Ing. Yosbel Falero Vento	**MSc. Yadier Perdomo Cuevas**
_____	_____
Firma del Tutor	Firma del Tutor

Agradecimientos

Antes que nada, quiero expresar mi más profundo agradecimiento a mis padres por hacer posible cumplir este gran sueño de convertirme en ingeniero. Su apoyo incondicional ha sido fundamental tanto en los momentos buenos como en los desafíos. No puedo dejar de mencionar a mi hermana pequeña Nico por estar siempre presente, aunque a veces muy traviesa, pero nunca deja de regalarme sonrisas y alegrar mis días.

Un agradecimiento especial a mi abuela Yaya, cuya fe en mí nunca ha flaqueado, y a mi tía Alina y mi primo Pedro, por su bondad constante. Aunque algunos familiares no puedan acompañarme hoy, su apoyo ha sido fundamental en mi camino; en particular, agradezco a mis primos jimaguas, Ernestico y Robertico, así como a mis tíos Lien y Titi, por su ayuda y aliento.

Finalmente, agradecer a mis amigos de la universidad, especialmente a Álvaro, por estar a mi lado durante este desafiante viaje y por toda la ayuda brindada. Su amistad y apoyo han sido pilares en esta etapa de mi vida. Gracias de corazón a todos.

Para concluir, quiero expresar mi gratitud hacia mí mismo por haber enfrentado este gran desafío y no haberme rendido ante las adversidades. Es un logro significativo que demuestra la fortaleza y la determinación que reside en cada uno de nosotros cuando nos comprometemos a superar los obstáculos que se presentan en nuestro camino.

Dedicatoria

Le quiero dedicar esta tesis en primer lugar a toda mi familia que ha estado ahí junto a mi en todo este viaje. A mis padres, mi hermana, mi abuela, mis primos, mis tias y tios. Ademas quiero dedicarle especialmente este logro a dos personas muy importantes que ya no estan, mi abuelo Nivaldo y mi tia Maritza. Gracias por todo.

Resumen

El presente trabajo tiene como objetivo desarrollar un sistema de recomendación que tenga en cuenta las preferencias de los usuarios y genere recomendaciones personalizadas para las plataformas del proyecto Z17 debido a las limitaciones de los sistemas homólogos actuales en las mismas. Para guiar el desarrollo se utilizó la metodología de desarrollo ágil AUP versión UCI escenario 2. Se generaron los artefactos fundamentales que propone la metodología para cada etapa de trabajo. Durante la investigación se analizaron los sistemas de recomendación existentes, así como las herramientas y técnicas para llevar a cabo el sistema. El estudio del estado del arte permitió identificar el algoritmo de recomendación, las funcionalidades y las tecnologías necesarias para implementar la solución propuesta. La solución es un sistema de recomendación basado en redes neuronales profundas, que utiliza modelos de Deep Learning para generar recomendaciones interesantes y personalizadas para los usuarios.

Palabras claves: Aprendizaje Profundo, Inteligencia Artificial, Proyecto Z17, Redes Neuronales, Sistema de recomendación.

Abstract

The objective of this work is to develop a recommendation system that considers user preferences and generates personalized recommendations for the platforms of the Z17 project, addressing the limitations of current analogous systems. The agile development methodology AUP version UCI scenario 2 was used to guide the development process. The fundamental artifacts proposed by the methodology were generated for each work stage. During the research, existing recommendation systems, tools, and techniques for system development were analyzed. The study of the state of the art allowed the identification of the recommendation algorithm, functionalities, and technologies required to implement the proposed solution. The solution is a recommendation system based on deep neural networks, utilizing Deep Learning models to generate personalized and engaging recommendations for users.

Keywords: *Deep Learning, Artificial Intelligence, Z17 Project, Neural Networks, Recommendation System.*

Índice General

INTRODUCCIÓN 11
CAPÍTULO 1: FUNDAMENTOS Y REFERENTES TEÓRICO-METODOLÓGICOS SOBRE EL OBJETO DE ESTUDIO. 16
1.1 Sistema de Recomendación 16
1.1.1 Clasificación de los Sistemas de Recomendaciones. 17
 Basados en Popularidad. 18
 Basados en Contenido. 18
 Basados en Filtrado Colaborativo. 19
 Filtrado Demográfico. 21
 Conversacionales. 21
 Híbridos. 22
1.1.2 Retroalimentación. 22
1.2 Deep Learning. 24
1.2.1 Sistemas de Recomendación basados en Deep Learning. 25
1.2.2 Ventajas de utilizar un sistema de recomendación basado en Redes Neuronales Profundas. 26
1.2.3 Desafíos. 27
1.3 Estudio de Sistemas Homólogos. 28
1.3.1 Estudio de Sistemas Homólogos en el ámbito Internacional. 29
1.3.2 Estudio de Sistemas Homólogos en el ámbito Nacional. 33
1.3.3 Conclusión de los sistemas homólogos. 36
1.4 Tecnologías y herramientas para el desarrollo. 38
1.4.1 Metodología de desarrollo de software. 38
1.4.2 Lenguaje de Programación. 39
Python V 3.11.9. 40
1.4.3 Base de datos. 40
MongoDB V 6.0.3. 40
1.4.4 Bibliotecas y Framerworks de Deep Learning. 41
Tensorflow V 2.15.0. 41
Keras V 2.15.0. 42
TensorFlow Recommenders V 0.7.3. 42
NumPy V 1.26.4. 43
Pandas V 2.2.3. 43
1.4.5 Entorno de Desarrollo. 43
Visual Studio Code V 1.92.2. 43
1.4.6 Servidor Web. 44
FastAPI V 0.115.0. 44
1.4.7 Herramientas de prueba. 44

JMeter V 5.6.3. 44
Acunetix V 11.0. 45
1.5 Conclusiones del Capítulo. 45
CAPÍTULO 2: ANÁLISIS Y DISEÑO DEL SISTEMA PARA LAS RECOMENDACIONES EN LAS PLATAFORMAS DEL PROYECTO Z17. 47
2.1 Descripción de la Propuesta de Solución. 47
2.1.1 Etapas. 47
2.1.2 Modelos. 49
2.2 Modelo Conceptual. 55
2.3 Diagrama de caso de uso del sistema. 57
2.4 Diagrama de clases de diseño. 58
2.5 Diagrama de secuencia. 59
2.6 Modelo de datos. 60
2.7 Requisitos de la propuesta de solución. 60
2.7.1 Requisitos Funcionales. 61
2.7.2 Requisitos no funcionales. 62
2.8 Arquitectura de software. 64
2.9 Patrones de Diseño. 66
2.9.1 Patrones GRASP. 66
 Experto. 66
 Creador. 67
 Controlador. 69
 Alta cohesión y bajo acoplamiento. 69
2.9.2 Patrones GOF. 71
 Singleton. 71
 Template Method. 72
2.9.3 Conclusiones de los patrones utilizados. 73
2.10 Conclusiones del capítulo. 74
CAPÍTULO 3: VALIDACIÓN Y PRUEBA DEL SISTEMA DE RECOMENDACIÓN BASADO EN DEEP LEARNING PARA LAS PLATAFORMAS DEL PROYECTO Z17. 75
3.1 Modelo de despliegue. 75
3.1.1 Descripción de elementos e interfaces de comunicación. 76
3.2 Estándares de codificación. 76
3.3 Estrategia de prueba. 77
3.3.1 Datos de entrenamiento. 78
3.3.2 Configuraciones del sistema. 80
3.4 Pruebas realizadas al sistema. 81
3.4.1 Prueba de Caja Negra. 81
3.4.2 Prueba de Caja Blanca. 85

3.4.3 Prueba de Rendimiento..... 88
3.4.4 Pruebas de Seguridad..... 90
3.5 Validación de la investigación. 92
3.6 Conclusiones del capítulo. 95
CONCLUSIONES GENERALES..... 96
RECOMENDACIONES 97
ANEXOS 98
Anexo 1: Preguntas concernientes a la entrevista realizada al cliente en el proceso de análisis e identificación de requisitos..... 98
Anexo 2: Validación mediante la técnica IADOV (Encuesta). 99
Anexo 3: Guía de Observación. 100
GLOSARIO DE TÉRMINOS..... 101
REFERENCIAS BILBIOGRÁFICAS..... 103

Índice de tablas

Tabla 1: Comparación entre SR Homólogos según las métricas utilizadas. 36
Tabla 2: Descripción de los Requisitos Funcionales. 61
Tabla 3: Descripción de los Requisitos no Funcionales. 62
Tabla 4: Descripción de los datos utilizados para entrenar los modelos. 78
Tabla 5: Configuraciones utilizadas para el proceso de entrenamiento de cada modelo. 81
Tabla 6: Cantidad de no conformidades por cada iteración de las pruebas funcionales. 82
Tabla 7: Caso de prueba Entrenar. 83
Tabla 8: Caso de prueba Generar Recomendaciones. 83
Tabla 9: Caso de prueba Actualizar. 84
Tabla 10: Caso de prueba para la prueba de caja blanca. 87
Tabla 11: Estadísticas resultantes de las pruebas de rendimiento utilizando el software JMeter. 89
Tabla 12: Resultados encontrados de las pruebas de seguridad realizadas. .. 91
Tabla 13: Cuadro lógico de IADOV. 92
Tabla 14: Índices de satisfacción. 93
Tabla 15: Resultados de la técnica IADOV. 93
Tabla 16: Variables de la formula ISG. 94

Índice de Figuras

Figura 1: Clasificación de Sistemas de Recomendación. ---- 17
Figura 2: Filtrado Basado en Contenido. ---- 19
Figura 3: Filtrado Colaborativo basado en usuarios . ---- 20
Figura 4: Filtrado Colaborativo basado en elementos . ---- 20
Figura 5: Clasificación de los Sistemas de Recomendación Conversacionales. ---- 21
Figura 6: Retroalimentación Explicita vs Implícita en sistemas de recomendaciones. ---- 24
Figura 7: Arquitectura de una red neuronal profunda llamada Modelo de dos torres para recomendación. ---- 26
Figura 8: Etapas del Sistema de Recomendación. ---- 47
Figura 9: Estructura de un sistema de recomendación de tres etapas. ---- 49
Figura 10: Formula del Producto Escalar o Dot Product. ---- 50
Figura 11: Arquitectura de una red neuronal de dos torres. ---- 50
Figura 12: Función de activación Softmax. ---- 51
Figura 13: Arquitectura del modelo de recuperación. ---- 53
Figura 14: Función de activación Sigmoidea. ---- 54
Figura 15: Arquitectura de un modelo de clasificación. ---- 54
Figura 16: Modelo Conceptual del contexto de la investigación ---- 55
Figura 17: Diagrama de caso de uso del sistema ---- 57
Figura 18: Diagrama de clases con estereotipos web. ---- 58
Figura 19: Diagrama de secuencia. ---- 59
Figura 20: Modelo de datos. ---- 60
Figura 21: Clase “RetrievalModel” la cual representa el modelo de recuperación donde se refleja el patrón Experto. ---- 67
Figura 22: Acción Entrenar encargada de crear instancias de modelos. ---- 68
Figura 23: Función controladora de la funcionalidad: “crear motor” donde se refleja el patrón controlador. ---- 69
Figura 24: Clase DataPipeline donde se refleja el patrón: Alta Cohesión. ---- 70
Figura 25: Clase ModelConfig la cual se utiliza para poner en práctica el patrón: Bajo Acoplamiento. ---- 70
Figura 26: Clase RetrievalStage donde se pone de manifiesto el patrón Singleton. ---- 71
Figura 27: Clase RankingStage donde se pone de manifiesto el patrón Singleton. ---- 71
Figura 28: Creación de dos instancias globales en el sistema. ---- 72
Figura 29: Clase interfaz que se utiliza como plantilla para la clase DataPipeline. ---- 73
Figura 30: Diagrama de despliegue. ---- 75
Figura 31: Comportamiento de las no conformidades de las pruebas funcionales ejecutadas. ---- 82
Figura 32: Función controladora para autenticar usuario. ---- 86
Figura 33: Técnica del camino básico. ---- 86

Introducción

En la era actual, caracterizada por la omnipresencia de las plataformas digitales, la creación de contenido ha experimentado un crecimiento exponencial, lo que ha resultado en una saturación de información en la red. Los usuarios, al navegar por internet, se encuentran con un volumen abrumador de datos, lo que dificulta la localización de contenido relevante y de calidad. Ante esta realidad, los sistemas de recomendación emergen como una solución esencial. Estos algoritmos inteligentes, diseñados con técnicas avanzadas de aprendizaje automático y análisis de datos, tienen la capacidad de filtrar y sugerir a los usuarios aquellos contenidos que más se alinean con sus intereses y comportamientos previos (Almaraz Pérez, 2013).

En Cuba ya existen plataformas que enfrentan problemas de exceso de información. Aquí es donde entra en juego el proyecto Z17, un equipo de jóvenes con una cultura ágil para el desarrollo de soluciones disruptivas e innovadoras, con un alcance global y enfocado en entregar productos de calidad a los cubanos (*Z17*, 2024).

Entre sus principales soluciones se encuentran (*Z17*, 2024):

- Apklis: Es el Centro Cubano de Aplicaciones Android enfocado en la distribución, actualización y comercialización de aplicaciones.
- toDus: La plataforma cubana de mensajería instantánea y colaborativa que permite el intercambio de mensajes, archivos y mucho más de forma inmediata.
- Picta: Una plataforma de contenido multimedia que permite la reproducción y transmisión en vivo.

Los sistemas de recomendación de las plataformas del proyecto Z17 enfrentan desafíos significativos, ya que no proporcionan una experiencia de usuario óptima. Esta situación se debe a su limitada capacidad para generar sugerencias personalizadas, su dificultad para adaptarse a las preferencias dinámicas de los usuarios y su ineficiencia al procesar grandes cantidades de datos. Estos

aspectos son cruciales para mejorar la interacción del usuario con la plataforma y garantizar su satisfacción. Esto limita la capacidad de las plataformas para mejorar la visibilidad de contenido menos conocido y adaptarse a las dinámicas cambiantes del mercado.

Las sugerencias personalizadas implican analizar el historial y las interacciones de los usuarios en las plataformas digitales. Esto permite recomendar contenido que sea de interés particular para cada individuo, en lugar de ofrecer únicamente contenido que podría ser considerado interesante de manera general. Estas sugerencias personalizadas son la problemática principal a resolver en este trabajo de investigación debido a su ausencia total dentro de los sistemas actuales de las plataformas del proyecto Z17.

La plataforma Picta utiliza un algoritmo que calcula la semejanza entre distintos contenidos. Para ello se basa en diferentes metadatos de los mismos tales como: nombre, actores, género y descripción. Además utiliza el historial de likes y dislikes de cada ítem para calcular la ponderación de cada uno. Este tipo de técnicas es conocida como sistema de recomendación basado en contenido.

Este proceso que realiza Picta para entrenar su algoritmo puede tardar hasta tres días para generar nuevos candidatos (Ver Glosario de Términos página 101). Esto es bastante lento y poco sostenible en el tiempo ya que al aumentar los datos de la plataforma el proceso de generar nuevas recomendaciones también aumenta considerablemente.

Actualmente se utilizan algoritmos convencionales basados en estadística lo cual es bastante útil para volúmenes de datos pequeños o medianos. Sin embargo, estas soluciones son insuficientes para plataformas que gestionan un contenido significativo y en constante crecimiento. Se habla de plataformas con millones de datos con una suma estimada de entre 50 y 100 millones de datos cada una.

Lo anterior conlleva a resolver el siguiente **problema**: ¿Cómo desarrollar un Sistema de Recomendación basado en Deep Learning que tenga en cuenta las preferencias de los usuarios y genere recomendaciones personalizadas para las plataformas del proyecto Z17?

Se plantea como **objetivo general**: Desarrollar un Sistema de Recomendación basado en Deep Learning que tenga en cuenta las preferencias de los usuarios y genere recomendaciones personalizadas para las plataformas del proyecto Z17.

Se establece como **campo de acción**: El Sistema de Recomendación para las plataformas del proyecto Z17.

Se define como **objeto de estudio**: Los Sistemas de Recomendación.

Para darle solución al problema, es necesario dar respuesta a las siguientes **preguntas científicas**:

1. ¿Cuáles son los principios teóricos y desarrollos recientes en la recomendación de información que sustenta la implementación de un sistema de recomendación basado en Deep Learning que tenga en cuenta las preferencias de los usuarios y genere recomendaciones personalizadas?

2. ¿Cómo implementar un sistema de recomendación basado en Deep Learning que tenga en cuenta las preferencias de los usuarios y genere recomendaciones personalizadas para las plataformas del proyecto Z17?

3. ¿Cómo evaluar a través de pruebas la calidad del sistema y de las recomendaciones para las plataformas del proyecto Z17?

Las **tareas de investigación** definidas para dar cumplimiento al objetivo de la investigación fueron las siguientes:

1. Elaboración del estado del arte de los sistemas de recomendación y los principales conceptos y elementos teóricos del tema.
2. Análisis y selección de las tecnologías, herramientas y metodologías de desarrollo necesarias para la implementación de la propuesta solución.
3. Análisis y diseño del proceso de ingeniería de software del Sistema de Recomendación basado en Deep Learning para las plataformas del proyecto Z17.

4. Implementación de las funcionalidades identificadas de la solución informática propuesta.
5. Aplicación de una estrategia de validación y prueba a partir de métodos y técnicas que demuestren la validez del resultado obtenido.

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

Métodos Teóricos:

Histórico-lógico: Este método se utilizó para analizar la evolución histórica de los sistemas de recomendación, identificando las etapas claves y los avances tecnológicos que han permitido su desarrollo. A partir de esta caracterización histórica, se concibió el sistema actual: un sistema de recomendación basado en Deep Learning. Este enfoque permitió comprender cómo los principios y técnicas históricas han influido en el diseño y la implementación de los sistemas modernos.

Analítico-sintético: El empleo de este método se evidencia cuando se realiza un análisis de toda la teoría y documentación, que permiten la extracción de los elementos fundamentales relacionados con el objeto de estudio.

Análisis documental: El uso de este método se realiza durante el desarrollo de la investigación. Facilita el estudio de documentos relacionados con la selección de los materiales de estudio y las técnicas de Inteligencia Artificial (IA). Sirve de referencia en la selección de materiales de estudio para la construcción de un sistema de recomendación.

Métodos Empíricos:

Entrevista: Empleado en el encuentro con el cliente para obtener información necesaria que permite determinar las características, cualidades y requisitos que debe poseer la propuesta solución (Ver Anexo 1).

Técnica IADOV: Este método se utiliza en la investigación para validar la retroalimentación de los usuarios respecto al nivel de satisfacción del SR (Sistema de Recomendación). A través de esta técnica, se recopilan y analizan las opiniones de los usuarios para evaluar la satisfacción y aceptación del SR propuesto.

La observación: permitió valorar las diferentes manifestaciones y comportamientos de los procesos y fenómenos relacionados con los sistemas de recomendación (Ver Anexo 3).

La presente investigación está conformada por la siguiente estructura: introducción, tres capítulos, conclusiones, recomendaciones, anexos, glosario de términos y referencias bibliográficas. Los capítulos abordan los siguientes temas:

Capítulo 1: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio. En este capítulo se realiza un análisis de los principales conceptos relacionados con el objeto de estudio, así como un análisis del estado del arte de los sistemas de recomendación. También se realiza un estudio de las distintas herramientas y tecnologías a utilizar en el desarrollo del sistema de recomendación propuesto.

Capítulo 2: Análisis y diseño del sistema para las recomendaciones en las plataformas del proyecto Z17. En este capítulo se determinan los servicios que brindará el sistema, definiéndose las funcionalidades que debe cumplir, así como el diseño e implementación del mismo, y se generan los artefactos que propone la metodología.

Capítulo 3: Validación y prueba del Sistema de Recomendación basado en Deep Learning para las plataformas del proyecto Z17. Este capítulo abarca todo lo relacionado con el diseño de los mecanismos utilizados para la verificación y validación de la solución propuesta. Se detallan también las pruebas que se le realizaron al sistema ya finalizado, con el objetivo de asegurar la eficiencia de la solución.

CAPÍTULO 1: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio.

En este capítulo, se realizará un recorrido histórico por la evolución de los sistemas de recomendaciones y sus diferentes algoritmos. Se analizarán las ventajas y limitaciones de cada enfoque así como una vista al uso de estos tanto en el ámbito nacional como internacional. Se elabora la fundamentación teórica de la investigación destacando los conceptos relacionados con el problema existente. Posteriormente, se presenta un análisis de las metodologías de desarrollo de software, herramientas, lenguajes y tecnologías necesarias para dar cumplimiento a las necesidades de la solución que se propone.

1.1 Sistema de Recomendación

Un sistema de recomendación está compuesto por algoritmos que se utilizan para sugerir productos o servicios al usuario. Su propósito principal es, a través de una serie de valoraciones y criterios sobre los datos del usuario o del ítem, predecir qué productos o servicios podrían ser del agrado del usuario, con el fin de mejorar su experiencia (Xia, 2023).

Los sistemas de recomendación se usan en una variedad de industrias, como por ejemplo la publicidad, la música, los libros, los juegos, las series y el cine. En el caso del cine, un sistema de recomendación de películas puede sugerir películas que se ajusten a los intereses y preferencias de los usuarios (Xia, 2023).

La siguiente lista muestra ejemplos de plataformas webs conocidas que necesitan sistemas de recomendación eficientes para mantener el interés de los usuarios debido a la vasta diversidad de contenidos disponibles (Casalegno, 2022).

1. **YouTube:** Cada minuto los usuarios suben **500 horas de vídeos** , es decir, un usuario tardaría 82 años en ver todos los vídeos subidos solo en la última hora.

2. **Spotify:** Los usuarios pueden escuchar más de **80 millones de canciones y podcasts** .

3. **Amazon:** Los usuarios pueden comprar más de **350 millones de productos diferentes** .

Todas estas plataformas utilizan potentes modelos de aprendizaje automático para generar recomendaciones relevantes para cada usuario (Casalegno, 2022).

En base a la información anterior se valora de que en las plataformas del proyecto Z17, es fundamental contar con un sistema de recomendación más personalizado y adaptable. Esto no solo mejora la visibilidad del contenido, sino que también retiene la atención de los usuarios, incrementando la popularidad de las plataformas y atrayendo a un mayor número de usuarios y clientes.

1.1.1 Clasificación de los Sistemas de Recomendaciones.

A lo largo de los años, el estudio en el campo de los sistemas de recomendaciones ha dado lugar a numerosos avances, culminando en el desarrollo de una variedad de algoritmos que ofrecen recomendaciones con gran precisión. A continuación, se emprenderá un recorrido por algunos de estos algoritmos, explorando sus principales características.

Existen diferentes tipos de técnicas que se pueden aplicar a un sistema de recomendación, como se puede observar en la Figura 1.

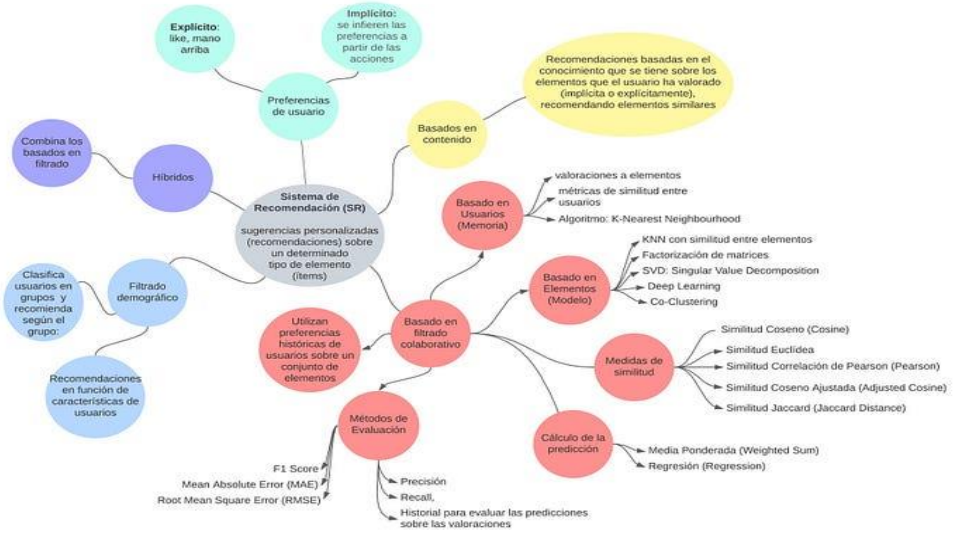


Figura 1: Clasificación de Sistemas de Recomendación (Gutierrez, 2023).

Los sistemas de recomendación se pueden clasificar en varias categorías. Estas categorías permiten entender mejor las características y aplicaciones de cada tipo de sistema. A continuación se presentaran las diferentes categorías que existen.

- **Basados en Popularidad.**

Los sistemas basados en la popularidad son implementados principalmente en las ventas de productos o sugerencias concretas. Estos toman como referencia la popularidad del objeto de estudio por una variable principal que puede ser el número de ventas, una característica especial o inclusive una oferta y se muestra de forma general a todos los usuarios que investiguen el área a la que pertenece el objeto. Estos sistemas suelen ser fáciles de implementar y gozan de cierto nivel de efectividad. Su desventaja principal es la imposibilidad de personalizar los criterios de sugerencia para el usuario (Xia, 2023).

Para el sistema de recomendación del proyecto Z17, se descartarán los métodos basados en la popularidad. Esta decisión se fundamenta en la prioridad de enriquecer la personalización dentro de las plataformas asociadas al proyecto. La personalización implica proporcionar sugerencias que sean específicas y ajustadas a las preferencias individuales de cada usuario, tomando como base su historial previo de interacciones. Por tanto la popularidad no es un factor importante en esta solución.

- **Basados en Contenido.**

Genera recomendaciones basadas en el conocimiento que se tiene sobre los elementos que el usuario ha valorado (implícita o explícitamente, Ver Epígrafe 1.1.2), recomendando elementos similares (Ver Figura 2). Este tipo de sistemas es uno de los que tiene mayor presencia en la actualidad. Con ellos podemos descubrir opciones que se ajusten a las características de los productos o contenidos que hemos disfrutado con anterioridad y elegir elementos similares (Zhang et al., 2021).

Figura 2: Filtrado Basado en Contenido (Xia, 2023).

Este enfoque se tendrá en cuenta y se aplicará en la propuesta solución, ya que facilita el uso de las características o metadatos de cada elemento. Esto enriquece al sistema de recomendación con un contexto más amplio y una información detallada de los datos, lo que incrementa la precisión. Además, permite recomendar elementos que no son necesariamente populares, utilizando sus características o metadatos específicos, lo cual contribuye a aumentar la visibilidad del contenido. Este enfoque es el utilizado actualmente en las plataformas Picta, toDus y Apklis.

- **Basados en Filtrado Colaborativo.**

El filtrado colaborativo también se trata de uno de los métodos más comunes en los sistemas de recomendación. Tiene dos enfoques, el filtrado colaborativo basado en usuarios y el filtrado colaborativo basado en elementos (Xia, 2023).

- **Sistemas basados en usuario** (memoria), El filtrado colaborativo basado en usuarios tiene como objetivo principal predecir los intereses de un usuario mediante la información que se le ha proporcionado sobre el historial, preferencias e información de muchos usuarios. La ventaja de este sistema es que es de fácil implementación y brindan un alto nivel de cobertura. La desventaja es que tendrá dificultades en recomendar productos nuevos, y usuarios nuevos, porque, debido a la falta de un historial, no se podrá sugerir recomendaciones personalizadas (Xia, 2023).

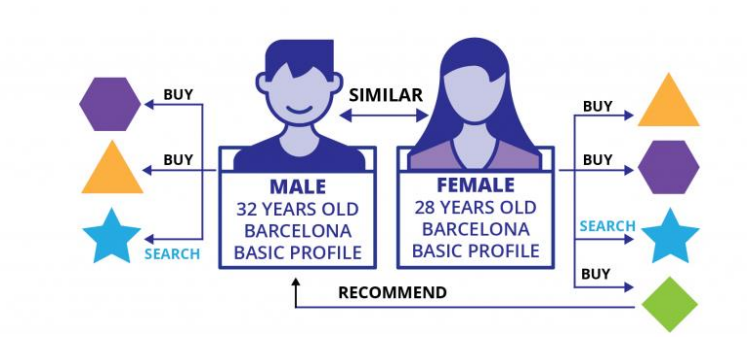


Figura 3: Filtrado Colaborativo basado en usuarios (Acosta, 2020).

- **Sistemas basados en elementos** (modelo), el filtrado colaborativo basado en elementos recomienda ítems basados en los ratings de los ítems realizados por otros usuarios en el sistema. La diferencia entre este filtrado y el anterior, es que no se realiza la búsqueda de vecindad de usuarios, sino que calcula directamente la similitud entre elementos (Xia, 2023).

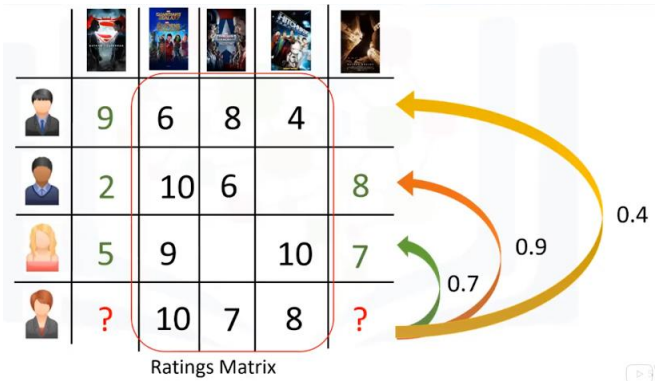


Figura 4: Filtrado Colaborativo basado en elementos (Xia, 2023).

En la propuesta solución se utilizará un enfoque que ha sido cuidadosamente considerado, destacando por su capacidad para personalizar y adaptar las recomendaciones a los usuarios. Esta característica representa una mejora significativa respecto a los sistemas existentes en las plataformas del proyecto Z17, donde la personalización aún no está presente. Para el desarrollo del sistema de solución, se optará por la primera variante del enfoque basado en filtrado colaborativo, conocido como Memory-based o basado en memoria. Este método se centra en el usuario y su historial de interacciones, permitiendo así que las recomendaciones sean personalizadas y relevantes para cada individuo.

- **Filtrado Demográfico.**

Estos sistemas clasifican a los usuarios en grupos y genera recomendaciones según el grupo al cual pertenece el usuario. Por ejemplo, hacer recomendación de sitios de interés según la ubicación geográfica del usuario, o recomendar elementos según la edad del usuario. Las recomendaciones se generan en función de las características de los usuarios (Alamdari et al., 2020).

Este método se excluirá de la propuesta solución debido a que no corresponde con los objetivos de esta investigación el hacer recomendaciones basadas en zonas geográficas o emplear filtros por localidades tales como municipios o provincias actualmente. Se enfoca la solución en las interacciones de los usuarios y el contenido de los elementos, dejando de lado la ubicación demográfica. No obstante, cabe destacar que este enfoque podría resultar interesante para considerarlo en futuras mejoras del proyecto.

- **Conversacionales.**

En este nuevo enfoque, los usuarios participan en un diálogo de recomendación, donde reciben recomendaciones y devuelven retroalimentación en forma de críticas sobre esas recomendaciones. De esta manera, permite al sistema refinar la búsqueda y ofrecer un conjunto de productos más adecuados a las preferencias del usuario (Xia, 2023). Estos sistemas guían al usuario a través del espacio de productos, ofreciendo sugerencias y solicitando retroalimentación. Existen varios tipos de SR conversacionales (Ver Figura 5):

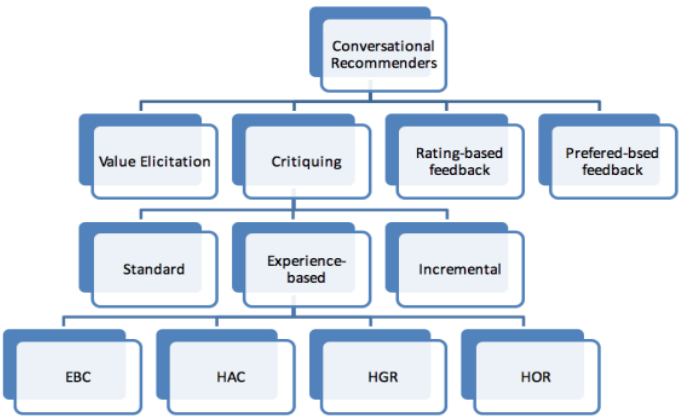


Figura 5: Clasificación de los Sistemas de Recomendación Conversacionales (Xia, 2023).

Este enfoque contribuiría a la personalización de las recomendaciones, un objetivo clave de la investigación. Sin embargo, su implementación implicaría una mayor complejidad en el desarrollo del sistema. Dadas las limitaciones de tiempo y los recursos del equipo de desarrollo, se ha decidido no adoptar este método. No obstante, se sugiere como una valiosa propuesta para mejoras futuras de la solución ya desarrollada.

- **Híbridos.**

Estos sistemas tratan de un modelo que combina diferentes enfoques con el objetivo de juntar sus mejores características y mejorar su rendimiento, de esta manera poder generar mejores recomendaciones (Varela et al., 2020).

Los sistemas de recomendación híbridos pueden dividirse en dos grupos (Varela et al., 2020):

- **De combinación lineal**: Son aquellos que crean una lista de recomendaciones sin combinarlas para crear una predicción combinada.
- **De combinación secuencial**: Donde la salida de una técnica de recomendación es la entrada a otra técnica.

En el sistema de recomendación a desarrollar para el proyecto Z17 se implementará una solución híbrida, perteneciente al grupo de combinación secuencial. Esta técnica integra las fortalezas del enfoque basado en contenido con las del filtrado colaborativo. Al hacerlo, se aprovecha las ventajas de ambos para producir recomendaciones de interés y de calidad, destinadas a enriquecer la experiencia de los usuarios en las plataformas asociadas al proyecto.

Este enfoque híbrido superará las limitaciones de los sistemas actuales en las plataformas como Picta, Apklis y toDus al aprovechar las preferencias de los usuarios aportando recomendaciones personalizadas.

1.1.2 Retroalimentación.

En el contexto de los sistemas de recomendaciones, el término retroalimentación se refiere a la información que los usuarios proporcionan, sobre su comportamiento o acciones con el contenido que consumen ya sea en una

plataforma web o en una aplicación móvil. Dicha retroalimentación puede categorizarse de manera implícita o explícita y es fundamental para esta investigación (Xie et al., 2020).

La retroalimentación implícita se recopila a través del comportamiento del usuario sin que este tenga que realizar acciones adicionales. En este caso el usuario no expresa sus intereses directamente lo que significa que son interacciones menos directas que contienen ruido. Con ruido se está refiriendo a datos poco precisos respecto a las preferencias de un usuario (Roy & Dutta, 2022).

Por otro lado, la retroalimentación explícita requiere una acción directa del usuario, donde este reacciona de alguna manera comunicando que le interesa o que no le interesa. Proporcionando información más exacta y de mejor calidad para un algoritmo de recomendación (Roy & Dutta, 2022).

Algunos ejemplos de recolección de datos de forma **explícitas** son (Roy & Dutta, 2022):

- Solicitar al usuario que pondere sobre la base de una escala proporcionada, algún tema en particular.
- Presentar al usuario dos temas, y solicitarle que seleccione uno de ellos.
- Solicitar al usuario que cree una lista de temas de su preferencia.

Algunos ejemplos de recolección de datos de forma **implícitas** son (Roy & Dutta, 2022):

- Guardar un registro de los temas que el usuario ha visto en una tienda en línea.
- Analizar el número de visitas que recibe un artículo.
- Guardar un registro de los artículos que el usuario ha seleccionado.

Figura 6: Retroalimentación Explicita vs Implícita en sistemas de recomendación (Casalegno, 2022).

En la propuesta de solución se tendrá en cuenta ambos tipos de retroalimentación: implícita y explícita. La combinación de ambas contribuye a la generación de recomendaciones personalizadas y precisas. Se utilizarán los datos implícitos para filtrar y recuperar elementos candidatos (Ver Glosario de Términos página 101) potencialmente relevantes de entre miles o millones disponibles. En contraste, los datos explícitos se emplearán para clasificar y priorizar estos elementos, asegurando que los candidatos presentados posean el mayor valor e interés para los usuarios.

1.2 Deep Learning.

El aprendizaje profundo o deep learning en inglés, se inspira en la estructura y función del cerebro humano, particularmente en cómo las neuronas se conectan y transmiten información. Este tipo de aprendizaje utiliza redes neuronales profundas para modelar y resolver problemas complejos, estas redes están compuestas por múltiples capas de nodos o neuronas interconectadas que simulan este proceso, permitiendo que la máquina 'aprenda' de los datos de entrada a través de la experiencia. Con cada nuevo dato procesado, la red ajusta sus pesos y sesgos (Ver Glosario de Términos página 101) internos para mejorar su precisión en la predicción de resultados, lo que resulta en un sistema que se vuelve más inteligente y eficiente con el tiempo (Sharifani & Amini, 2023).

1.2.1 Sistemas de Recomendación basados en Deep Learning.

Los sistemas de recomendación basados en redes neuronales representan una de las aplicaciones más innovadoras de la inteligencia artificial en la era del Big Data.

El estudio *"Deep Neural Networks for YouTube Recommendations"* (Covington et al., 2016) representa un hito en el desarrollo de sistemas de recomendaciones siendo la base científica y tecnológica en la que se basan los potentes sistemas de recomendación que utiliza Google en sus productos estrellas como YouTube y Google Play Store (Covington et al., 2016). A continuación, se ofrecerá una visión del funcionamiento de estos sistemas y como han transformado el campo de las recomendaciones personalizadas. Además se trataran algunos desafíos importantes que traen consigo el utilizar este tipo de tecnología.

La Figura 7 muestra la estructura de un modelo de Deep Learning conocido como Modelo de Dos Torres. Este modelo es usado en sistemas de recomendación para la recuperación de información. Su nombre se debe a que la arquitectura tiene la apariencia de dos torres conectadas. Cada torre procesa un tipo distinto de datos: una torre representa al usuario o contexto, y la otra representa los elementos candidatos (Ver Glosario de Términos página 101), como videos o aplicaciones. La conexión entre ambas torres se realiza mediante el producto escalar de los vectores de salida de cada una, lo que produce el resultado final de la recomendación.

Las capas ocultas en cada torre ayudan a **aprender representaciones abstractas** de los datos para mejorar la precisión de las recomendaciones. Este diseño permite que el sistema comprenda tanto las preferencias del usuario como las características de los elementos (Kammoun et al., 2022).

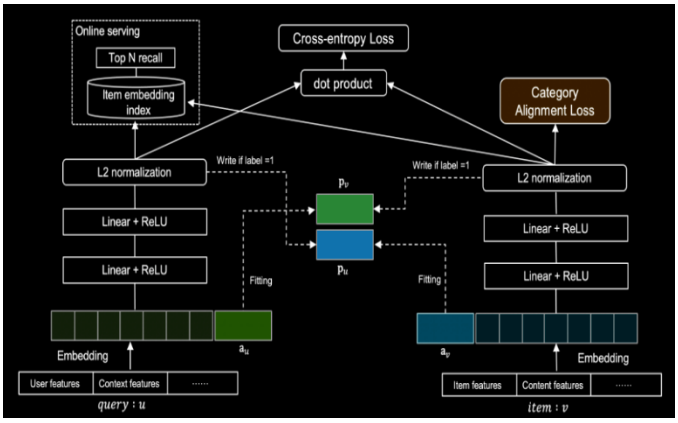


Figura 7: Arquitectura de una red neuronal profunda llamada Modelo de dos torres para recomendación (Kammoun et al., 2022).

Este modelo tiene un enfoque híbrido ya que utiliza filtrado basado en contenido (Ver Epígrafe 1.1.1) procesando características de los datos como nombres, descripción, edad de un usuario, categoría de un producto, etc. Además utiliza un filtrado colaborativo tanto basado en usuario como basado en elementos (Ver Epígrafe 1.1.1) ya que dependiendo de con que datos se entrene puede aprender a relacionar usuarios y elementos en base a su historial de interacciones (Kammoun et al., 2022). Por tanto se ha seleccionado este modelo para ser utilizado como parte del sistema de recomendación a desarrollar en esta investigación. A continuación se trataran algunas ventajas que tiene utilizar este tipo de modelos de Deep Learning en el campo de las recomendaciones.

1.2.2 Ventajas de utilizar un sistema de recomendación basado en Redes Neuronales Profundas.

Estas herramientas avanzadas no solo no se limitan a analizar grandes volúmenes de datos, sino que también son capaces de discernir y aprender patrones complejos y no lineales que serían imposibles de detectar por métodos estadísticos tradicionales. Utilizando algoritmos de redes neuronales profundas, estos sistemas pueden procesar y cruzar una variedad de factores, desde el comportamiento de navegación en la web hasta las preferencias de compras y las interacciones sociales, para entregar recomendaciones personalizadas y mejorar así la experiencia del usuario (Betru et al., 2017).

Una ventaja significativa de estos sistemas es su capacidad para realizar aprendizaje automático sin supervisión. Esto significa que pueden identificar

patrones y correlaciones sin necesidad de etiquetado previo, reduciendo la necesidad de intervención humana y permitiendo descubrimientos que podrían pasar desapercibidos incluso para expertos en el tema (Sharifani & Amini, 2023).

Además, la capacidad de procesar y aprender de las interacciones de los usuarios de forma ágil y rápida utilizando Fine Tunning (Ver Glosario de Términos página 101) puede reducir significativamente los tiempos de cálculo, abordando el problema de los sistemas actuales que pueden tardar días en actualizarse con nuevos datos. Esto no solo mejorará la experiencia del usuario, sino que también hará que las plataformas sean más ágiles y capaces de adaptarse rápidamente.

La creación de un sistema de recomendaciones avanzado para las plataformas del proyecto Z17 representa un desafío significativo y una oportunidad para mejorar la interacción del usuario. El uso de un sistema de recomendación basado en Deep Learning en plataformas como Apklis, toDus y Picta puede transformar la experiencia del usuario al proporcionar sugerencias más interesantes y personalizadas.

1.2.3 Desafíos.

Desarrollar un sistema de recomendaciones utilizando redes neuronales profundas para las plataformas Picta, toDus y Apklis presenta una serie de desafíos únicos y complejos. Uno de los principales retos es la necesidad de grandes volúmenes de datos de alta calidad para entrenar el modelo, además, la arquitectura de las redes neuronales profundas es intrínsecamente compleja y requiere una cuidadosa selección de hiperparámetros (Ver Glosario de Términos página 101), lo que puede llevar a un proceso de prueba y error que consume mucho tiempo. (Gao et al., 2023).

La tendencia al sobreajuste es otro problema común; las redes neuronales profundas pueden aprender patrones irrelevantes de los datos de entrenamiento que no se generalizan bien a datos no vistos. Esto puede llevar a recomendaciones que no son aplicables o útiles para el usuario. Además, la equidad y el sesgo son preocupaciones significativas, ya que los sistemas de

recomendación pueden perpetuar o incluso amplificar sesgos existentes en los datos de entrenamiento (J. Chen et al., 2023).

Pueden existir sesgos como (J. Chen et al., 2023):

- **Sesgo de popularidad:** el sistema tiende a recomendar elementos que ya son populares, ignorando opciones menos conocidas.
- **Sesgo de diversidad**: se tiende a recomendar siempre lo mismo sin tener en cuenta otros elementos que no ha visto el usuario pero le pueden ser de interés.
- **Sesgo de interacción**: Este sesgo se produce cuando las interacciones de los usuarios con el sistema (como clics, valoraciones, etc.) no reflejan sus verdaderas preferencias.
- **Sesgo de Temporalidad:** Este sesgo se relaciona con la evolución de las preferencias de los usuarios a lo largo del tiempo. Un modelo que no se actualiza regularmente puede volverse obsoleto.

Los sesgos se refieren a las tendencias o predisposiciones que se introducen en los sistemas de inteligencia artificial durante su entrenamiento, lo que puede llevar a resultados parciales, discriminatorios o injustos. Estos sesgos pueden afectar la precisión, la confiabilidad y la imparcialidad de los sistemas de inteligencia artificial (J. Chen et al., 2023).

Los sesgos pueden ser un gran desafío tanto para encontrarlos como para solucionarlos por lo que se debe de tener mucho cuidado con los datos de entrenamiento y los hiperparámetros (Ver Glosario de Términos página 101) de los modelos a utilizar. Los hiperparámetros son configuraciones ajustables que se definen antes de entrenar el modelo y afectan cómo aprende (Sharifani & Amini, 2023).

1.3 Estudio de Sistemas Homólogos.

En la presente investigación, se han analizado diversos sistemas de recomendación implementados a nivel nacional e internacional, con el objetivo de identificar sus puntos fuertes y áreas de mejora. Para esto se tuvieron en cuenta algunos criterios como:

- **Preferencias de usuarios:** Determina si se tuvieron en cuenta las preferencias de los usuarios para hacer la recomendación.
- **Algoritmo híbrido:** La capacidad de utilizar dos o más algoritmos en el sistema analizado.
- **Personalización:** Este criterio mide qué tan bien las recomendaciones se ajustan a las preferencias y comportamientos individuales de los usuarios.
- **Deep Learning:** Evalúa si la tecnología utilizada está basada en redes neuronales profundas o lo que es lo mismo en Deep Learning.

Con esta información, se está en proceso de adaptar y refinar el propio sistema de recomendación, asegurando que se beneficie de las ventajas observadas y evite las desventajas detectadas, para ofrecer una experiencia de usuario superior y más personalizada.

1.3.1 Estudio de Sistemas Homólogos en el ámbito Internacional.

- **Sistema de Recomendación de YouTube.**

YouTube es la plataforma más grande del mundo para crear, compartir y descubrir contenido de video por lo que representa uno de los sistemas de recomendación industrial de mayor escala y más sofisticados que existen. Las recomendaciones de YouTube son responsables de ayudar a más de mil millones de usuarios descubriendo contenido personalizado de un corpus en constante crecimiento de videos (Covington et al., 2016).

Entre las funciones que realiza el sistema de recomendación de Youtube se puede encontrar (Mata Arias, 2017):

- Tiene en cuenta las preferencias de los usuarios para hacer la recomendación.
- Emplea la recomendación personalizada.
- Establece relaciones de gustos similares entre los usuarios.
- Además de la personalización se utiliza la regionalización (recomendar criterios basados en el área geográfica).
- Posee una vista donde se muestran y gestionan las recomendaciones.

En combinación con otras áreas de productos de Google, YouTube ha experimentado un cambio de paradigma fundamental hacia el uso del aprendizaje profundo como una solución de propósito general para casi todos los problemas de aprendizaje (Covington et al., 2016).

Pros y contras:

Este sistema facilita una comprensión más profunda de la solución que se quiere desarrollar. Provee conocimientos esenciales para implementar y desarrollar el sistema deseado en esta investigación dando a ver elementos como: recomendaciones personalizadas, retroalimentación y como comparativa para mejorar la solución propuesta. Sin embargo, no se empleó este sistema de manera exacta debido a que, es un software propietario, costoso y está diseñado para operar a nivel global lo que no resulta adecuado para soluciones más específicas y particulares.

- **Sistema de Recomendación de Netflix.**

Los sistemas de recomendación de Netflix abarcan varios enfoques algorítmicos como el aprendizaje por refuerzo, las redes neuronales, los modelos causales, los modelos gráficos probabilísticos y la factorización matricial (Steck et al., 2021).

Cada vez que se accede al servicio de Netflix, su sistema de recomendación intenta ayudar al usuario a encontrar fácilmente una serie, una película o un videojuego de su agrado. Para calcular la probabilidad de que le gustaría un determinado título del catálogo, se basan en varios factores, entre ellos (Mata Arias, 2017):

- La interacción con los servicios (como el historial de visualización y las calificaciones asignadas a otros títulos).
- Actividad de otros miembros con gustos y preferencias similares.
- Información sobre los títulos, como género, categorías, actores, año de lanzamiento, etc.

Además de saber qué el usuario ha visto en Netflix, también tienen en cuenta otros factores para personalizar las recomendaciones. Algunos de ellos son (Mata Arias, 2017):

- A qué hora del día el usuario accede a Netflix.
- Qué idiomas prefiere.
- Con qué dispositivos accede a Netflix.
- Cuánto tiempo le dedica a un título de Netflix.

Todos estos datos son parte de la información con la que alimentan sus algoritmos. El sistema de recomendaciones no incluye información demográfica (como la edad o el género) en el proceso de toma de decisiones (Steck et al., 2021).

Pros y contras:

Este sistema funciona sobre Netflix, el cual presenta notables similitudes con la plataforma Picta, analizada previamente en la investigación. Ambos gestionan contenidos audiovisuales, incluyendo películas, series y documentales. Esta comparación es fundamental para entender cómo se puede implementar un sistema de recomendación en plataformas similares. Identificando los tipos de retroalimentación posibles y los diversos factores que contribuyen a la personalización de las recomendaciones, tales como el idioma y el tiempo de visualización. Sin embargo, es importante destacar que se trata de un sistema cerrado y propietario, además de ser muy complejo lo que puede obstaculizar su comprensión y la adaptación a necesidades más específicas.

- **Sistema de Recomendación de Google Play Store.**

El sistema de recomendación de Google Play Store representa un hito en la ingeniería de sistemas de inteligencia artificial aplicada al campo de las recomendaciones personalizadas. La personalización es un pilar fundamental de este sistema, permitiendo que las sugerencias de aplicaciones y juegos sean únicas para cada usuario, basándose en su historial de interacciones previas. Esto se logra mediante el uso de modelos avanzados de aprendizaje automático que incluyen generadores de candidatos (Ver Glosario de Términos página 101),

que son capaces de procesar y evaluar más de un millón de aplicaciones para identificar aquellas que mejor se alinean con las preferencias del usuario (Gong & Zhernov, 2024).

Para mejorar aún más el rendimiento del sistema, Google ha implementado técnicas de vanguardia como la transición de modelos LSTM a Transformers, que son más eficientes en la captura de dependencias a largo plazo entre los elementos. La atención aditiva eficiente es otra mejora significativa que reduce los costos computacionales sin sacrificar la calidad de las recomendaciones. La ponderación por importancia es una técnica diseñada para mitigar los sesgos (Ver Glosario de Términos página 101) en las recomendaciones, asegurando que las sugerencias sean justas y equitativas. (Gong & Zhernov, 2024).

Pros y contras:

Este sistema opera en una interfaz similar a la de Apklis; ambas funcionan como tiendas de aplicaciones móviles. Esto proporciona un marco de referencia para comprender cómo implementar un sistema de recomendación en dichas plataformas. Se analiza la retroalimentación de los usuarios y se exploran métodos de recomendación, como el basado en contenido o el filtrado colaborativo. Además, incorpora tecnología de redes neuronales profundas, que también se empleará en la solución propuesta. Dado que es un sistema de gran envergadura que funciona a nivel global y cuenta con miles de millones de usuarios, su magnitud excede el alcance de esta investigación. Esta complejidad lo hace inadecuado para aplicaciones de menor escala como las de este estudio.

Conclusiones de los sistemas homólogos internacionales.

Se eligieron los sistemas de recomendaciones internacionales análogos de plataformas líderes como YouTube, Netflix y Google Play Store ya que revelan aspectos fundamentales para el desarrollo de un sistema de recomendación basado en Deep Learning en el marco del proyecto Z17. Entre las ventajas, se destaca la posibilidad de realizar benchmarking (Ver Glosario de Términos página 101) y adoptar mejores prácticas, lo que permite establecer comparativas con los estándares de la industria.

Sin embargo, existen desventajas notables al utilizar estos sistemas como la complejidad técnica, que implica un profundo entendimiento de áreas especializadas como el aprendizaje automático y el manejo de grandes volúmenes de datos. Los requerimientos de recursos son también un desafío considerable, dado que la implementación de sistemas de recomendación demanda una infraestructura robusta y costosa.

1.3.2 Estudio de Sistemas Homólogos en el ámbito Nacional.

- **Subsistema de recomendación de información para el buscador cubano Orión.**

Actualmente internet supone una de las principales fuentes de información que existe. Por este motivo es que en los últimos tiempos se han desarrollado alternativas para la recuperación de esta información como es el caso del buscador cubano Orión. En el presente sistema homologo se desarrolla un Subsistema de recomendación de información con el objetivo de mejorar la calidad en los resultados mostrados a los usuarios por este buscador, teniendo en cuenta sus preferencias y necesidades individuales (Mata Arias, 2017).

Como principales tecnologías empleadas se encuentran Solr como mecanismo de indexación, Symfony como marco de trabajo PHP, PHP como lenguaje de programación y NGINX como servidor de aplicación (Mata Arias, 2017).

Pros y contras:

Este subsistema adopta un enfoque híbrido que integra el filtrado colaborativo basado en usuarios con el filtrado basado en contenido (Mata Arias, 2017). Esta combinación ha servido como guía para profundizar en el entendimiento de estos métodos, los cuales también se aplicarán en la solución propuesta por esta investigación. Sin embargo, presenta algunas limitaciones debido al uso de algoritmos tradicionales o estadísticos, lo que resulta contrario para los objetivos de este estudio. En su lugar, se busca implementar redes neuronales profundas o Deep Learning para mejorar la precisión y eficacia de los algoritmos utilizados.

- **Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0.**

Este sistema es el encargado de gestionar todas las producciones científicas e investigativas generadas en la universidad de las ciencias informáticas (UCI). El cúmulo de información existente en REPXOS 3.0 provoca que los usuarios al realizar las búsquedas no siempre obtengan como resultado en primera posición los ítems más relevantes de acuerdo a sus preferencias y se dificulte la localización de dichos ítems. Para solucionar este problema, se concibió el desarrollo del Módulo de Recomendaciones que contribuyó a la recuperación de los ítems más relevantes para sugerirlos a los usuarios del sistema.

Está definido como un sistema de recomendación basado en filtrado colaborativo con un enfoque basado en memoria y de este el esquema Usuario-Usuario (Ver Epígrafe 1.1.1). Se obtuvo como resultado un módulo para el sistema REPXOS 3.0 que realiza recomendaciones automáticas y manuales de ítems a usuarios. (Ruiz Ricardo & Vaillant Valdéz, 2015).

Pros y contras:

En este homologo se tuvo en cuenta el estudio profundo de un sistema de recomendaciones colaborativo, un enfoque de filtrado colaborativo basado en memoria con un esquema Usuario-Usuario el cual aporto bases de conocimiento y guía para el mismo enfoque que se utilizará en la presente investigación. Sin embargo, este sistema no se implementó como solución debido a que no ofrece una alternativa híbrida, elemento clave para mejorar la precisión de un sistema de recomendación. Además, está diseñado para gestionar contenidos de producción científica e investigativa, lo cual difiere significativamente del tipo de contenido presente en las plataformas objetivo de esta investigación, como Picta, toDus y Apklis. Por tanto, su adaptación resultaría en un proceso complejo y oneroso.

- **Desarrollo del Sistema de Recomendación de equipos de investigación para tesis de grado.**

Este trabajo abarca un sistema de recomendación como apoyo al proceso de conformación de equipos de investigación, y que favorezca un mejor desarrollo del trabajo de diploma de los estudiantes de quinto año en la Universidad de las Ciencias Informáticas (UCI). Cabe destacar que el sistema no pretende

conformar los equipos de investigación sino ser una herramienta útil en la toma de decisiones basada en el análisis de datos y en técnicas de inteligencia artificial empleadas a la hora de realizar las recomendaciones (Iglesias Mizrahi, 2016).

Para la implementación de esta solución se utilizó un SR híbrido con características principales de los Sistemas de Filtrado Colaborativo. El sistema utilizará la recomendación de los resúmenes estadísticos, pero utilizando el filtrado basado en contenido fundamentalmente, aunque se trata de un SR híbrido. En este se empleó inteligencia artificial para realizar el filtrado y realizar las recomendaciones (Iglesias Mizrahi, 2016).

Pros y contras:

El sistema homólogo analizado se centra en el método de filtrado basado en contenido, lo que lo convierte en un referente para comprender mejor esta técnica, la cual se aplicará en la solución propuesta por esta investigación. Es importante señalar que este sistema incorpora un enfoque híbrido. No obstante, se decidió no adoptarlo debido a su dependencia de algoritmos probabilísticos y tradicionales, los cuales presentan un enfoque distinto que resulta poco compatible con los objetivos específicos de este estudio, relacionados con el proyecto Z17 y sus plataformas asociadas.

Conclusiones de los sistemas homólogos nacionales.

El análisis de los sistemas similares nacionales ha sido invaluable para extraer aprendizaje clave sobre la implementación de sistemas de recomendación. No han sido utilizados debido a la divergencia tecnológica entre los algoritmos tradicionales, tanto probabilísticos como estadísticos, y los algoritmos de redes neuronales profundas que se emplearán en esta investigación. El análisis de estos homólogos nacionales ha servido como un marco de referencia esencial para comprender el funcionamiento de los sistemas de recomendación híbridos, que combinan técnicas de filtrado colaborativo y basado en contenido, enfoques adoptados en el presente estudio.

1.3.3 Conclusión de los sistemas homólogos.

Luego de analizar los sistemas de recomendación existentes, tanto a nivel internacional como nacional, se concluye que ninguno de ellos satisface completamente los requisitos específicos de la investigación. Los sistemas nacionales analizados utilizan algoritmos y técnicas tradicionales o estadísticas como los métodos de regresión lineal, modelos de similitud del coseno y árboles de decisiones. Adecuados para sus propósitos originales pero insuficientes para abordar el desafío presente: plataformas con abundante contenido y crecimiento constante.

Los sistemas similares internacionales estudiados suelen requerir licencias costosas y limitan la personalización (Ver Glosario de Términos página 101) y adaptación a necesidades particulares. Además, muchos de estos sistemas son propietarios, lo que restringe el acceso al código fuente y dificulta su integración con otras herramientas o la implementación de mejoras.

Tabla 1: Comparación entre SR Homólogos según las métricas utilizadas (Fuente: Elaboración propia).

Sistemas	Personalización	Preferencias de usuarios	Algoritmo híbrido	Deep Learning
Sistema de Recomendación de Youtube.	Se tiene en cuenta.	Se tiene en cuenta.	Se utiliza.	Se utiliza.
Sistema de Recomendación de Netflix.	Se tiene en cuenta.	Se tiene en cuenta.	Se utiliza.	Se utiliza.
Sistema de Recomendación de Google Play Store.	Se tiene en cuenta.	Se tiene en cuenta.	Se utiliza.	Se utiliza.
Subsistema de recomendación de información para	Se tiene en cuenta.	Se tiene en cuenta.	Se utiliza.	No se utiliza.

el buscador cubano Orión.				
Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0.	Se tiene en cuenta.	Se tiene en cuenta.	No se utiliza.	No se utiliza.
Desarrollo del Sistema de Recomendación de equipos de investigación para tesis de grado.	Se tiene en cuenta.	Se tiene en cuenta.	Se utiliza.	No se utiliza.

La propuesta de solución busca desarrollar un sistema de recomendación que tenga las siguientes características:

- **Personalizable y dinámico:** permitiendo modificaciones y adaptaciones según las necesidades específicas de cada dominio.
- **Nacional:** un sistema soberano tecnológicamente del que no dependa de ninguna entidad extranjera para su uso.
- **Recomendaciones personalizadas:** Genere recomendaciones para cada usuario en base a sus intereses previos.
- **Tiene en cuenta las preferencias de los usuarios:** Se deben de estudiar y analizar las preferencias previas de los usuarios para en base a estas crear las recomendaciones.

La falta de sistemas de recomendaciones de código abierto y personalizables representa una barrera para la innovación en diversos sectores, como el comercio electrónico, el entretenimiento y la educación. Al desarrollar un sistema propio, se promueve la independencia tecnológica y se allana el camino para la creación de soluciones más eficientes y adaptadas a las necesidades locales.

1.4 Tecnologías y herramientas para el desarrollo.

Las herramientas informáticas son programas, aplicaciones o simplemente instrucciones usadas para efectuar tareas de modo más sencillo (Yanover, 2016). Con el objetivo de minimizar los costos, se propone utilizar tecnologías y herramientas que permitan su uso sin necesidad de pago de licencias.

1.4.1 Metodología de desarrollo de software.

En la adecuada implementación de la solución propuesta, se seleccionó la metodología AUP-UCI (UCI, sigla referente a la Universidad de Ciencias Informáticas). Esta metodología constituye una variante de AUP (Proceso Unificado Ágil, por sus siglas en inglés), y surge con el objetivo de ser una metodología que se adapte al ciclo de vida definido por la actividad productiva en la universidad. Ofrece prácticas que se centran en el desarrollo de productos y servicios de calidad (González Matos, 2021).

Para implementar esta estrategia de trabajo, se tuvieron en cuenta diversos aspectos relacionados con la naturaleza de la situación planteada y la solución propuesta. La metodología AUP-UCI ha sido adaptada específicamente para los proyectos desarrollados en la UCI, respondiendo tanto a las necesidades particulares del proyecto como a las del equipo de desarrollo. Esta metodología estructura el proceso de desarrollo de software en tres fases bien definidas: Inicio, Ejecución y Cierre. En particular, la fase de Inicio permite identificar de manera clara los objetivos y las estrategias, proporcionando un marco organizado para la planificación y ejecución del proyecto.

Un componente clave se manifiesta claramente durante la etapa de modelado del negocio, proporcionando una comprensión más profunda de los procesos y requerimientos del proyecto Z17. Asimismo, se fomenta la calidad del software a través de ajustes y mejoras ágiles que responden a la necesidad de mejorar los servicios ofrecidos. Como aspecto fundamental, la metodología pone un énfasis significativo en la relevancia de las pruebas y la validación a lo largo de todo el proceso de desarrollo.

En la variación de la metodología AUP-UCI, existen tres formas de encapsular los requerimientos: Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requerimientos por proceso (DRP), agrupados en cuatro escenarios, quedando de la siguiente forma:

Escenario No 1: Proyectos que modelen el negocio con Casos de Uso del Negocio (CUN) solo pueden modelar el sistema con CUS.
Escenario No 2: Proyectos que modelen el negocio con Modelo Conceptual (MC) solo pueden modelar el sistema con CUS.
Escenario No 3: Proyectos que modelen el negocio con DPN (Descripción de proceso de negocio) solo pueden modelar el sistema con DRP.
Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU (Historial de usuario).

Teniendo en cuenta los escenarios de la variación AUP-UCI, se decide encapsular los requisitos en el escenario número 2, debido a que el contexto seleccionado se aplica a 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. Además, el escenario elegido posibilita adaptar los requisitos mediante la retroalimentación constante con el cliente, lo cual es crucial para asegurar el acceso de calidad a los servicios ofrecidos por el sistema de recomendación a desarrollar.

Entre otros aspectos, se incluyen la implementación de prácticas ágiles que facilitan la creación de contenido de calidad. En adición, se fomenta el desarrollo colaborativo e iterativo, y se identifican y corrigen problemas antes de la implementación final.

1.4.2 Lenguaje de Programación.

Un lenguaje de programación según Gervacio (2018) consiste en un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Estos se utilizan para crear programas que controlen el comportamiento físico y lógico de una máquina y para expresar algoritmos con precisión. A continuación, se describe el lenguaje de programación a utilizar en el desarrollo de la propuesta de solución:

Python V 3.11.9.

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma («Python», 2024).

Python ofrece una sintaxis clara y legible que facilita la lectura y escritura de código. Su sencillez reduce considerablemente el tiempo de desarrollo. Cuenta con una biblioteca estándar extensa que cubre diversas áreas, incluyendo machine learning, manipulación de datos, estadística y visualización. Esto proporciona herramientas robustas para trabajar con los grandes conjuntos de datos típicos en sistemas de recomendaciones. Cuenta con una comunidad enorme y activa que proporciona abundante documentación, tutoriales y recursos en línea (Alfaro & Roberto, 2022).

La elección de este lenguaje de programación se debe principalmente a la experiencia que posee el equipo de desarrollo en su uso, así como a la extensa gama de bibliotecas y Frameworks disponibles en Python para la creación de proyectos de aprendizaje profundo e inteligencia artificial. Esta riqueza de recursos, unida a la facilidad de uso y versatilidad, sitúa a Python en una posición de liderazgo en comparación con otros lenguajes en el ámbito de la inteligencia artificial convirtiéndolo en la mejor opción para esta investigación.

1.4.3 Base de datos.

MongoDB V 6.0.3.

MongoDB es una base de datos NoSQL de tipo documental que almacena datos en un formato similar a JSON, llamado BSON (*Binary JSON*). A diferencia de las bases de datos relacionales tradicionales, MongoDB no utiliza tablas ni filas, sino que almacena la información en documentos organizados en colecciones. Esto permite que los datos sean más flexibles y escalables (*MongoDB*, 2024).

El sistema de base de datos elegido se ha seleccionado cuidadosamente para manejar la complejidad de los datos implicados en la propuesta de solución, particularmente en lo que respecta a las configuraciones de los modelos. Estas configuraciones requieren el manejo de un volumen significativo de datos no estructurados de diversas índoles, lo que representa un desafío considerable para su almacenamiento y gestión en una base de datos relacional. La capacidad de este sistema para gestionar eficientemente dicha complejidad es fundamental para el éxito de la implementación propuesta.

1.4.4 Bibliotecas y Framerworks de Deep Learning.

En informática, una biblioteca o librería, llamada por vicio del lenguaje, son un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca. Un Framerwork es una estructura o plataforma de desarrollo predefinida que proporciona un conjunto de herramientas, librerías y pautas que facilitan la creación y organización de aplicaciones (Tsui et al., 2022). A continuación, se describen las bibliotecas y Framerworks a utilizar en el desarrollo de la propuesta de solución:

Tensorflow V 2.15.0.

TensorFlow es un framerwork de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos (*TensorFlow*, 2024).

TensorFlow permite construir una amplia variedad de modelos de recomendación, desde los más simples (como filtrado colaborativo basado en matrices) hasta los más complejos (redes neuronales profundas con arquitecturas personalizadas). Esta flexibilidad es crucial para adaptarse a diferentes tipos de datos y requisitos de negocio (*TensorFlow*, 2024).

TensorFlow está diseñado para manejar grandes conjuntos de datos y modelos complejos. Puede ejecutarse en una sola CPU, múltiples GPUs o incluso en

clústeres de máquinas, lo que lo hace adecuado para sistemas de recomendaciones a gran escala (*TensorFlow*, 2024).

Esta biblioteca fue seleccionada principalmente porque constituye la fundación sobre la cual se construyen otras bibliotecas como Keras o Tensorflow Recommenders. Representa el nivel más elemental en la jerarquía de abstracción de las bibliotecas empleadas, siendo la de nivel más básico. Además permite construir y manjar modelos de Deep Learning eficientemente.

Keras V 2.15.0.

Keras es una biblioteca de Redes Neuronales de Código abierto escrita en Python. Es capaz de ejecutarse sobre TensorFlow. Está especialmente diseñada para posibilitar la experimentación en poco tiempo con redes de aprendizaje profundo. Sus fuertes se centran en ser amigable para el usuario, modular y extensible («Keras», 2024).

La biblioteca Keras ha sido elegida por su excepcional capacidad para simplificar la gestión de modelos de redes neuronales profundas. Ofrece una abstracción significativamente más elevada en comparación con TensorFlow, lo que facilita y agiliza el proceso de desarrollo. Esta característica la convierte en una herramienta valiosa para desarrollar un sistema de recomendaciones basado en redes neuronales profundas.

TensorFlow Recommenders V 0.7.3.

TensorFlow Recommenders (TFRS) es una biblioteca creada por Google para compilar modelos de sistemas de recomendaciones. Facilita el flujo de trabajo completo de la compilación de sistemas de recomendación: preparación de datos, formulación de modelos, entrenamiento, evaluación e implementación. Se basa en Keras y se enfoca en lograr una curva de aprendizaje suave manteniendo flexibilidad para compilar modelos complejos (*TensorFlow Recommenders*, 2024).

La biblioteca TensorFlow Recommenders de Google ha sido elegida por su amplia gama de herramientas prácticas para la creación de sistemas de recomendación mediante redes neuronales. Ofrece algoritmos avanzados para

la búsqueda de los vecinos más cercanos y métricas robustas para la evaluación de modelos. Además, simplifica el desarrollo, aumentando la eficiencia y reduciendo la curva de aprendizaje en proyectos de inteligencia artificial.

NumPy V 1.26.4.

NumPy es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas (*NumPy - About Us*, 2024).

La biblioteca Numpy ha sido seleccionada por su excelente compatibilidad con otras bibliotecas en uso, siendo extremadamente beneficiosa para ejecutar operaciones matemáticas complejas y eficientes en extensos conjuntos de datos. Además, cuenta con el respaldo de una vasta y activa comunidad.

Pandas V 2.2.3.

Pandas es una librería de Python especializada en la manipulación y el análisis de datos. Ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales, es como el Excel de Python. Es un software libre distribuido bajo la licencia BSD (Berkeley Software Distribution) (*Pandas*, 2024).

Se seleccionó la biblioteca Pandas debido a su amplia comunidad de usuarios y su capacidad para manejar y procesar datos de manera eficiente. Además, ofrece una excelente compatibilidad con otras bibliotecas utilizadas en el proyecto, lo que facilita la integración y el desarrollo de soluciones robustas.

1.4.5 Entorno de Desarrollo.

Visual Studio Code V 1.92.2.

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, MacOS y Linux. Viene con soporte integrado para Python y tiene un rico ecosistema de extensiones

para otros lenguajes (como C ++, C #, Java, Javascript, PHP, Go) y tiempos de ejecución (como .NET y Unity) (Microsoft, 2024).

Este editor fue seleccionado para el desarrollo de este proyecto ya que aporta una buena facilidad de uso y gran cantidad de extensiones útiles para agilizar el proceso de desarrollo.

1.4.6 Servidor Web.

FastAPI V 0.115.0.

FastAPI es un moderno y rápido Framework web para la creación de servidores webs con Python, que se basa en las pistas de tipo estándar de Python. Sus características claves incluyen un rendimiento muy alto, comparable con NodeJS y Go, lo que lo convierte en uno de los Frameworks de Python más rápidos disponibles. Además, FastAPI facilita la codificación ya que es muy flexible e intuitivo. Su diseño fácil de usar y aprender minimiza el tiempo dedicado a leer documentación (*FastAPI*, 2024).

Este Framework web ha sido elegido para el desarrollo del servidor debido a su notable flexibilidad y facilidad de uso en la programación. Está escrito en el mismo lenguaje que se utilizará para el sistema de recomendación, Python, lo que garantiza una integración fluida. Además, es uno de los servidores web más veloces disponibles, un aspecto crucial para proyectos de inteligencia artificial donde el rendimiento representa un reto significativo.

1.4.7 Herramientas de prueba.

JMeter V 5.6.3.

Apache JMeter es una herramienta de código abierto desarrollada por la Apache Software Foundation. Se utiliza principalmente para realizar pruebas de carga y medir el rendimiento de aplicaciones web, servicios web, bases de datos y otros recursos de servidor (*Apache JMeter - Apache JMeter™*, 2024).

Se ha seleccionado Apache JMeter para las pruebas de rendimiento de esta solución porque es una herramienta de código abierto, gratuita y muy versátil. Soporta múltiples protocolos como HTTP, HTTPS, REST y JDBC, lo que lo hace

ideal para probar tanto aplicaciones web como APIs (*Apache JMeter - Apache JMeter™*, 2024). Su interfaz gráfica facilita la configuración de pruebas, pero también permite personalización avanzada con scripts. Aunque existen alternativas como Gatling o K6, JMeter destaca por su comunidad activa, soporte de plugins y facilidad de uso, sin necesidad de altos costos o licencias.

Acunetix V 11.0.

Acunetix es una herramienta de software especializada en la seguridad de aplicaciones web. Es un escáner automatizado que detecta vulnerabilidades en sitios web, aplicaciones y APIs, ayudando a protegerlas contra amenazas como inyección SQL, XSS (cross-site scripting) y configuraciones inseguras (*Acunetix | Web Application and API Security Scanner*, 2024).

Se ha utilizado esta herramienta ya que es ideal para realizar pruebas de seguridad en aplicaciones gracias a su precisión y automatización avanzada. Es capaz de identificar vulnerabilidades críticas como inyección SQL y XSS, además de escanear APIs (Ver Glosario de Términos página 101) y aplicaciones modernas como SPAs (Single Page Applications o Aplicaciones de una sola página) (*Acunetix | Web Application and API Security Scanner*, 2024). Comparado con otras herramientas, como Burp Suite u OWASP ZAP, destaca por su facilidad de uso, rapidez y capacidad para manejar tecnologías avanzadas sin necesidad de configuraciones complejas.

1.5 Conclusiones del Capítulo.

Luego del estudio y análisis realizado del objeto de investigación del presente trabajo de diploma, apoyado en los métodos de la investigación científicos definidos, se concluye lo siguiente:

- ✓ Se realizó un estudio sobre los principales conceptos asociados al estudio de la presente investigación y las relaciones entre ellos. Esto permitió alcanzar un mayor dominio sobre los fundamentos de los Sistemas de Recomendación basados en redes neuronales profundas.

- ✓ La bibliografía consultada tanto a nivel nacional como internacional aportó que la definición y características del objeto de estudio se centran en autores de países foráneos y por consiguiente los sistemas similares existentes abundan mayormente en países extranjeros. Pero, destacar que los sistemas similares sirvieron como guía para entender el funcionamiento de las principales características que conformarán el sistema de recomendación de esta investigación como los enfoques que utiliza, la personalización y la tecnología de Deep Learning.

- ✓ El análisis del caso de estudio: Sistema de recomendación basado en Deep Learning para las plataformas del proyecto Z17, proporcionó una oportunidad única para identificar con precisión las características distintivas de la propuesta solución. Estas características son: robustez, Deep Learning y personalización. Además, se pudieron establecer claramente las ventajas que ofrece en comparación con otras opciones disponibles, así como las posibles desventajas o limitaciones inherentes a la misma.

- ✓ El ecosistema de Python, con bibliotecas como TensorFlow, Keras y TensorFlow Recommenders, proporciona un entorno de desarrollo ágil y eficiente para la investigación y experimentación con diferentes algoritmos de recomendación. La integración de estas herramientas con MongoDB y el Framerwork web FastAPI permitirá almacenar, procesar datos y utilizar el sistema como servicio para realizar las integraciones con las plataformas Picta, toDus y Apklis, lo que es fundamental para el entrenamiento de modelos de aprendizaje automático complejos.

CAPÍTULO 2: Análisis y diseño del sistema para las recomendaciones en las plataformas del proyecto Z17.

En este capítulo, a partir del estudio de los procesos del negocio, sus descripciones y su modelado, se describe el sistema a desarrollar. Se obtienen los artefactos relacionados a la ingeniería de software aplicada a la propuesta de solución tomando como punto de partida el problema de investigación. Además, se plasman los requisitos funcionales y no funcionales de la propuesta, así como los diferentes artefactos relacionados con la metodología de desarrollo.

2.1 Descripción de la Propuesta de Solución.

A continuación se hará una detallada explicación de cómo está compuesto el sistema propuesto. Para ello el autor se basó en las etapas que lo forman y los modelos que lo integran.

2.1.1 Etapas.

El sistema está compuesto por **3 etapas** fundamentales, **Generación de candidatos, Clasificación y Re-clasificación**.

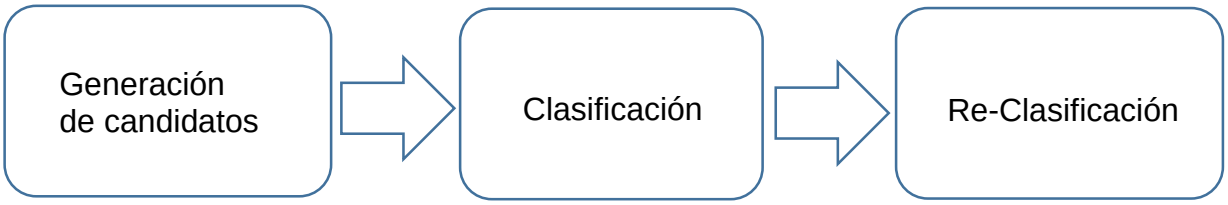


Figura 8: Etapas del Sistema de Recomendación (Fuente: Elaboración propia).

Estas etapas están conectadas entre sí de manera secuencial respectivamente donde la salida de una es la entrada de la siguiente. La última etapa es la encargada de devolver los datos a recomendar y la primera la encargada de obtener el corpus de elementos de candidatos inicial. A continuación cada una de las etapas:

- **Generación de candidatos**: Esta etapa es responsable de seleccionar un conjunto inicial de cientos o miles de candidatos entre el corpus base de

elementos que pueden ser millones. El objetivo principal de esta etapa es eliminar de manera eficiente a todos los candidatos que no les interesan al usuario. Dado que se puede estar tratando con millones de candidatos, tiene que ser computacionalmente eficiente. Esta etapa está compuesta por un modelo de Deep Learning el cual es entrenado con datos implícitos (véase epígrafe 1.1.2) para realizar el proceso de recuperación o generación de candidatos (Gao et al., 2023).

- **Clasificación**: Toma los candidatos obtenidos por la etapa anterior (Generación de candidatos) y los ajusta para seleccionar el mejor puñado de recomendaciones posibles basándose en los datos explícitos (véase el epígrafe 1.1.2) que tenga disponible. Su tarea es reducir el conjunto de elementos que pueden interesar al usuario a una lista corta de posibles candidatos que ronde entre los cientos. La idea de esta etapa es obtener candidatos más preciso que en la etapa anterior ya que al basarse en los datos explícitos tiene información de mayor valor sobre que le interesa o no al usuario. Esta etapa también está compuesta por un modelo de deep learning al igual que la anterior para realizar el proceso de clasificación (Gao et al., 2023).

- **Re-clasificación:** En la etapa final el sistema puede volver a clasificar para considerar criterios o limitaciones adicionales. Estos son algunos de los diversos factores que pueden ayudar a mejorar considerablemente las recomendaciones de un sistema. Los factores que tiene en cuenta esta etapa son los siguientes (Gao et al., 2023):

    1. **La actualidad** vela por que los elementos sean recientes y no generar candidatos antiguos.

    2. **La diversidad** valida que los candidatos sean diversos según las preferencias del usuario, todas las recomendaciones no pueden ser igual a lo que el usuario ve, ya que esto impide la visibilidad del contenido, algo fundamental en un sistema de recomendación.

3. **La equidad** evita sesgos (Ver Glosario de Términos página 101) en las recomendaciones como podrían ser sebos de clics o falsas tendencias y permite que todos los usuarios sean tratados de manera justa.

La Figura 9 muestra visualmente todo el proceso de etapas explicado anteriormente (Covington et al., 2016; Gao et al., 2023).

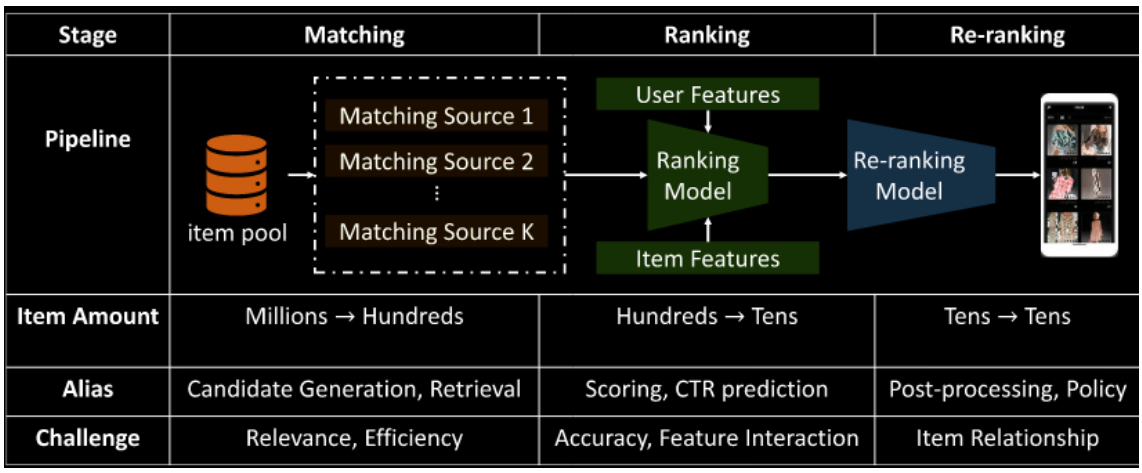


Figura 9: Estructura de un sistema de recomendación de tres etapas (Gao et al., 2023).

2.1.2 Modelos.

Existen tres tipos de modelos utilizados en este sistema, Modelos de Dos Torres, Modelo de Recuperación y Modelo de Clasificación (Covington et al., 2016):

Modelo de Dos Torres: es una arquitectura de red neuronal diseñada específicamente para tareas de aprendizaje de representaciones y, en particular, para problemas de recomendación, búsqueda y clasificación. Su nombre proviene de la estructura visual que adopta: “dos torres" de redes neuronales que procesan independientemente dos tipos de datos, y luego combinan sus representaciones finales para obtener una puntuación o predicción (Kammoun et al., 2022).

En una arquitectura de dos torres, cada torre es una red neuronal que procesa características de entrada candidatas o de consulta para producir una representación integrada de esas características. Debido a que las

representaciones de incrustación son simplemente vectores de la misma longitud, se puede calcular el producto escalar entre estos dos vectores para determinar qué tan cerca están. Esto significa que la orientación del espacio de incrustación está determinada por el producto escalar de cada par <query, candidato> en los ejemplos de entrenamiento (Kammoun et al., 2022).

Donde “query” representa a los usuarios o la información de contexto y “candidato” las características de los elementos que se quieren recomendar, como películas, series, aplicaciones, etc (Kammoun et al., 2022).

La fórmula para calcular el producto escalar es la siguiente:

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

Donde:

- a = primer vector
- b = segundo vector
- n = dimensión del espacio vectorial
- a_i = componente del vector a
- b_i = componente del vector b

Figura 10: Formula del Producto Escalar o Dot Product (Kammoun et al., 2022).

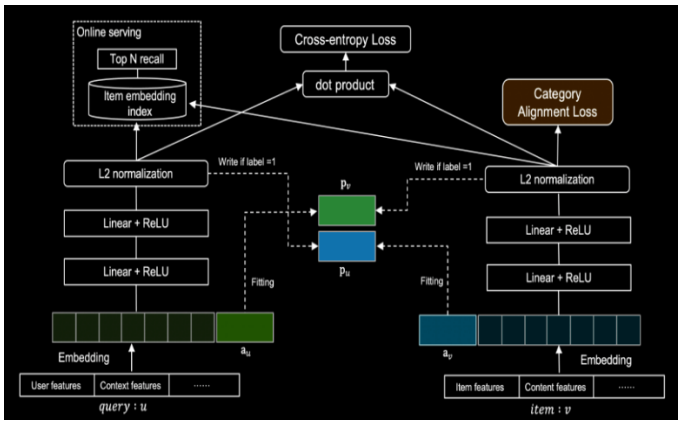


Figura 11: Arquitectura de una red neuronal de dos torres (Kammoun et al., 2022).

Ambas son técnicas de recuperación basadas en incrustaciones que calculan representaciones vectoriales de consultas y candidatos de dimensiones inferiores, donde la similitud entre estos dos vectores se determina calculando

su producto escalar (Kammoun et al., 2022). En esta investigación se utiliza este modelo de dos torres como base de los 2 modelos siguientes. Esto quiere decir que los modelos a continuación están formados dentro de su arquitectura por un modelo de dos torres. Gracias a esto son capaces de transformar los datos y crear una representación vectorial para cada una de las interacciones que contienen los datos de entrenamiento.

Modelo de Recuperación: Es el encargado de leer el corpus de datos completo y recuperar candidatos más probables a ser de interés para el usuario. Este tipo de modelos es entrenado con datos implícitos (véase el epígrafe 1.1.2) del usuario (Covington et al., 2016).

- ✓ **Representación**: Los modelos de recuperación a menudo se componen de dos sub modelos:
  - Un modelo de consulta que calcula la representación de la consulta (normalmente un vector de incorporación de dimensionalidad fija) mediante funciones de consulta.
  - Un modelo candidato que calcula la representación candidata (un vector de igual tamaño) utilizando las características candidatas.

  Estos dos sub modelos son representados con el modelo de Dos Torres explicado anteriormente.

- ✓ **Función de activación de la capa de salida**: Se utiliza una capa de salida con función de activación Softmax (Figura 12) que calcula la probabilidad de interés de un usuario hacia cada candidato (Covington et al., 2016).

$$P(k) = \frac{e^{s_k}}{\sum_{j=1}^{K} e^{s_j}}$$

Figura 12: Función de activación Softmax (Tomás Cruz, 2024).

Donde:
P(k) es la probabilidad de que la entrada pertenezca a la categoría "k".
s_k es la puntuación asociada a la categoría "k".
K es el número total de categorías.

- ✓ **Métrica y función de coste**: Para medir la precisión se utiliza una métrica llamada **FactorizedTopK**, diseñada para evaluar el desempeño de modelos de recomendación. Esta métrica se enfoca en medir la capacidad del modelo para predecir correctamente los elementos (productos, películas, etc.) más relevantes para un usuario dado en una lista ordenada de recomendaciones (Covington et al., 2016).

  Recibe la lista de candidatos donde para cada consulta (usuario) se obtiene la lista de candidatos predichos por el modelo y se compara esta lista con la lista real de candidatos seleccionados por el usuario. Luego se calcula la precisión para diferentes valores de k (generalmente 1, 5, 10, 50 y 100). La precisión se define como el porcentaje de candidatos predichos que coinciden con los candidatos reales (Covington et al., 2016).

- ✓ **Servicio**: Este modelo una vez entrenado es explotado para construir un servicio eficiente mediante la construcción de un índice aproximado de vecinos más cercanos (ANN) el cual se utiliza en producción para obtener los n candidatos en la etapa de recuperación (Covington et al., 2016).

La Figura 13 a continuación muestra gráficamente la arquitectura de un modelo de recuperación de Deep Dearning donde su funcionamiento fue explicado anteriormente. Este modelo será utilizado en la propuesta de solución en la etapa de generación de candidatos.

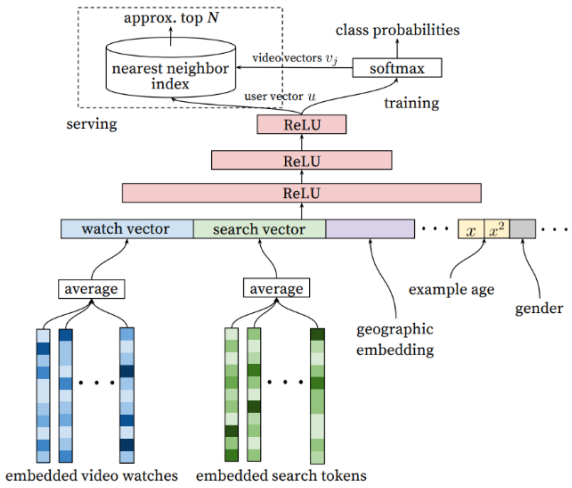


Figura 13: Arquitectura del modelo de recuperación (Covington et al., 2016).

El modelo de recuperación desempeña un papel fundamental en la fase inicial del proceso de entrenamiento del sistema. Este paso preliminar es crucial para integrar las interacciones implícitas de los usuarios, tal como se detalla en el epígrafe 1.1.2. En el contexto del proyecto Z17, se analizan acciones como los clics en videos de plataformas como Picta o de aplicaciones en Apklis. La importancia de este modelo radica en su capacidad para aprender de los patrones de comportamiento de los usuarios, lo que a su vez permite personalizar la experiencia al adaptarse a sus preferencias individuales.

Modelo de Clasificación: Después de la generación de candidatos, otro modelo califica y clasifica los candidatos para obtener así los más interesantes para los usuarios. Este modelo se utiliza en la etapa de clasificación del sistema y se entrena con datos explícitos (Ver Epígrafe 1.1.2) para realizar la clasificación (Covington et al., 2016).

- ✓ **Representación**: Utiliza la misma representación del modelo anterior (Modelo de Recuperación).

- ✓ **Función de activación de la capa de salida**: Se utiliza una capa de salida con función de activación Sigmoidea (Figura 14) que calcula la probabilidad de que ocurra una de las dos clases utilizadas. Estas, por ejemplo, en el caso particular de la plataforma Picta son: un like (1) o un dislike (0) (Kyurkchiev & Markov, 2015).

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Figura 14: Función de activación Sigmoidea (Kyurkchiev & Markov, 2015).

Donde:

σ(z): Es el valor de salida de la función Sigmoide.

z: Es el valor de entrada a la función Sigmoide.

e: Constante matemática Euler igual a 2.71828.

- ✓ **Métrica y función de coste**: Para medir la precisión se utiliza una métrica llamada **Binary Accuracy** o Precisión Binaria específica diseñada para evaluar el desempeño de modelos de clasificación binaria (Kyurkchiev & Markov, 2015). Esta métrica se utiliza en el modelo de clasificación para clasificar candidatos en base a likes y dislikes dados por los usuarios.

- ✓ **Servicio**: Este modelo una vez entrenado es compilado y almacenado para ser utilizado posteriormente como parte del sistema de recomendación en producción.

La Figura 15 a continuación muestra gráficamente la arquitectura de un modelo de clasificación de Deep Learning donde su funcionamiento fue explicado anteriormente. Este modelo será utilizado para la propuesta de solución en la etapa de clasificación.

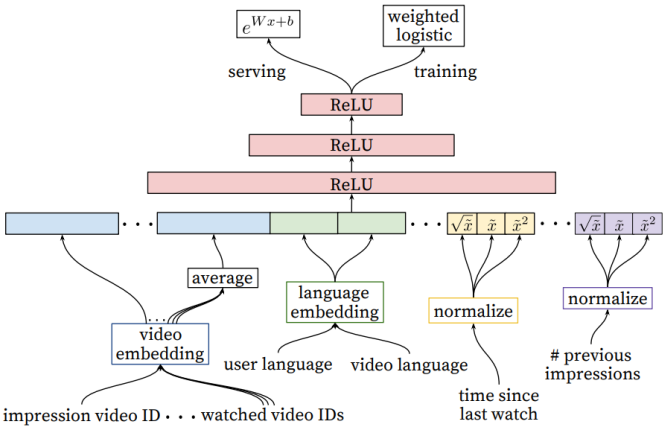


Figura 15: Arquitectura de un modelo de clasificación (Covington et al., 2016).

El modelo de clasificación es utilizado en la fase intermedia del sistema. Esta fase es crucial para integrar las interacciones explicitas de los usuarios, tal como se detalla en el epígrafe 1.1.2. En el contexto del proyecto Z17, se analizan acciones como los likes y dislikes en videos de plataformas como Picta. La importancia de este modelo radica en aprender preferencias de los usuarios interesantes y de calidad para aportar precisión al sistema de recomendación.

Conocer y entender cómo funciona el sistema a implementar forma parte del proceso de desarrollo y es un paso importante en esta investigación. Luego de haber analizado en profundidad cómo funciona el sistema de recomendación que se propone implementar en este estudio entonces se continuara con el modelado de los artefactos propuestos por la metodología de desarrollo utilizada (AUP-UCI Escenario 2).

2.2 Modelo Conceptual.

Un modelo conceptual es un artefacto de la disciplina de análisis, construido con las reglas UML. Tiene como objetivo comprender y describir las clases más importantes, así como, identificar y explicar los conceptos significativos en el dominio del problema, identificando los atributos y las asociaciones existentes entre ellos (González Matos, 2021; SOMMERVILLE, 2005).

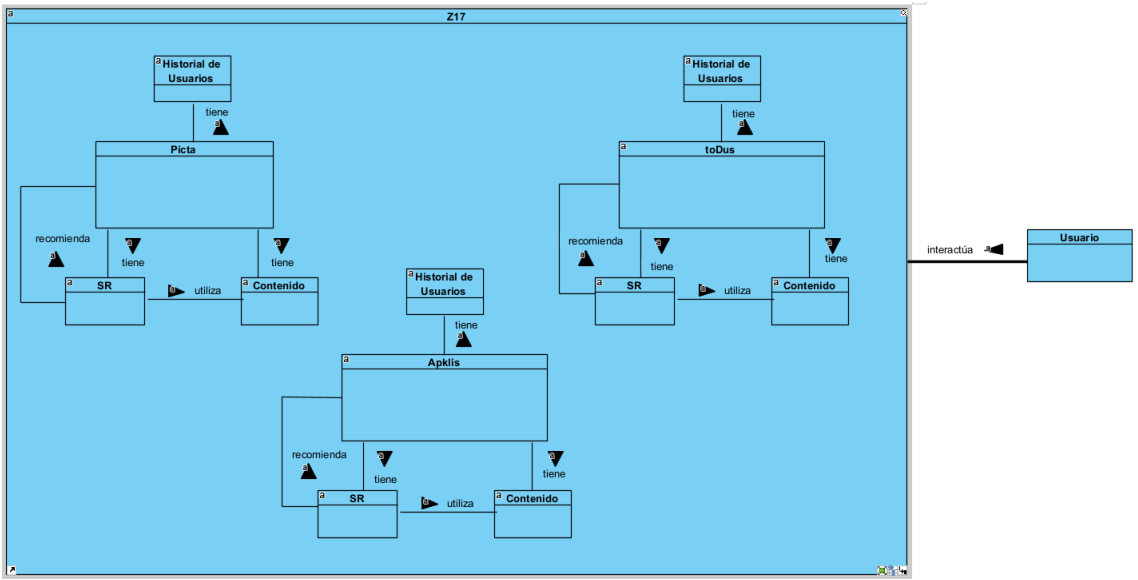


Figura 16: Modelo Conceptual del contexto de la investigación (Fuente: Elaboración propia).

- **Proyecto Z17**: Es un proyecto que engloba varias soluciones de software integradas.
- **Picta**: Es una plataforma de contenido multimedia. Permite a los usuarios acceder y disfrutar de videos, música y otros tipos de medios digitales.
- **toDus**: Es una plataforma de mensajería instantánea. Los usuarios pueden enviar y recibir mensajes de texto, imágenes, videos y otros contenidos en tiempo real.
- **Apklis**: Es una plataforma de distribución de aplicaciones Android. Los desarrolladores pueden publicar sus aplicaciones, y los usuarios pueden buscarlas, descargarlas e instalarlas.
- **Contenido**: Se refiere a todo el material digital disponible en las plataformas. Esto incluye videos en Picta, canales en toDus y aplicaciones en Apklis.
- **Historial de Usuarios**: Son las acciones o comportamientos que han realizado los usuarios previamente en cada plataforma.
- **Usuarios**: Son las personas que interactúan con las diferentes plataformas. Realizan acciones como consumir contenido, comunicarse y descargar aplicaciones.
- **SR (Sistema de Recomendación)**: Es el sistema encargado de generar las recomendaciones por los usuarios. Analiza el contenido de cada plataforma respectivamente para generar recomendaciones a los usuarios.

El modelo conceptual (Figura 16) contiene en primer lugar el proyecto Z17 y las plataformas que lo conforman Apklis, Picta y toDus. Cada plataforma tiene su propio contenido digital, el historial de sus usuarios y un sistema de recomendación interno. Los usuarios de estas plataformas interactúan con las mismas. Los sistemas de recomendaciones utilizan el contenido de cada plataforma respectivamente para generar recomendaciones a las mismas. Este diagrama permite entender y tener una visión más general de cómo funciona el modelo de negocio en el proyecto Z17 y sus plataformas para el contexto de esta investigación.

En el modelo conceptual (Figura 16), se observa que los sistemas de recomendación actuales no utilizan el historial de los usuarios en las plataformas. Esto contrasta notablemente con el sistema de recomendación desarrollado en

esta investigación, ya que este sí incorpora el historial de comportamientos y acciones previas de los usuarios. Al hacerlo, es capaz de generar recomendaciones personalizadas que consideran las preferencias individuales de cada usuario.

2.3 Diagrama de caso de uso del sistema.

Los diagramas de caso de uso son una técnica para capturar requisitos o información de cómo un sistema o negocio trabaja, y están compuesto por los casos de uso, los actores que se pueden definir como algo con comportamiento, como una persona (identificada por un rol), sistema informatizado u organización, y las relaciones existentes entre ambos (González Matos, 2021).

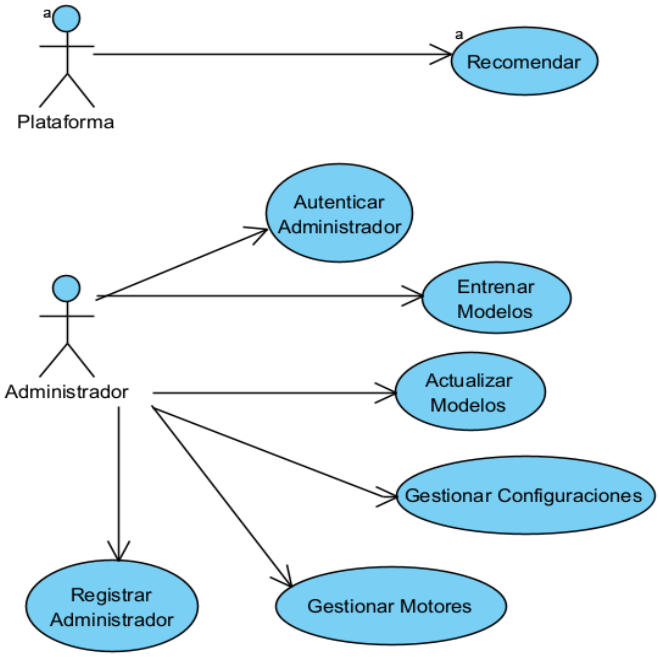


Figura 17: Diagrama de caso de uso del sistema

(Fuente: Elaboración propia).

El diagrama consta de dos actores, Plataforma, la cuál será la única encargada con el rol de generar las recomendaciones para que sus usuarios las vean. Por otro lado está el rol de administrador, el cual será el encargado de administrar el sistema de recomendación en su totalidad, puede entrenar los modelos, actualizar los modelos, gestionar las configuraciones de los modelos y gestionar los motores. Se le llama motor a cada instancia de un sistema de recomendación. Se pueden crear instancias de estos sistemas ya que son 3 plataformas y cada una va a utilizar una instancia o un motor. Por último, el administrador también

puede autenticarse y registrar nuevos administradores. Estas son las funcionalidades que tendrá el sistema a desarrollar.

Los casos de usos gestionar están compuestos de 4 métodos fundamentalmente, crear, leer, actualizar y eliminar (CRUD: Create, Read, Update y Delete). En el caso de gestionar motores, se incluye leer un motor para mostrar sus datos, actualizar sus datos, eliminar el motor y crear nuevos motores.

2.4 Diagrama de clases de diseño.

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces de software en una aplicación. A diferencia de las clases conceptuales del Modelo del Dominio, las clases de diseño de los DCD muestran las definiciones de las clases software en lugar de los conceptos del mundo real (González Matos, 2021; SOMMERVILLE, 2005).

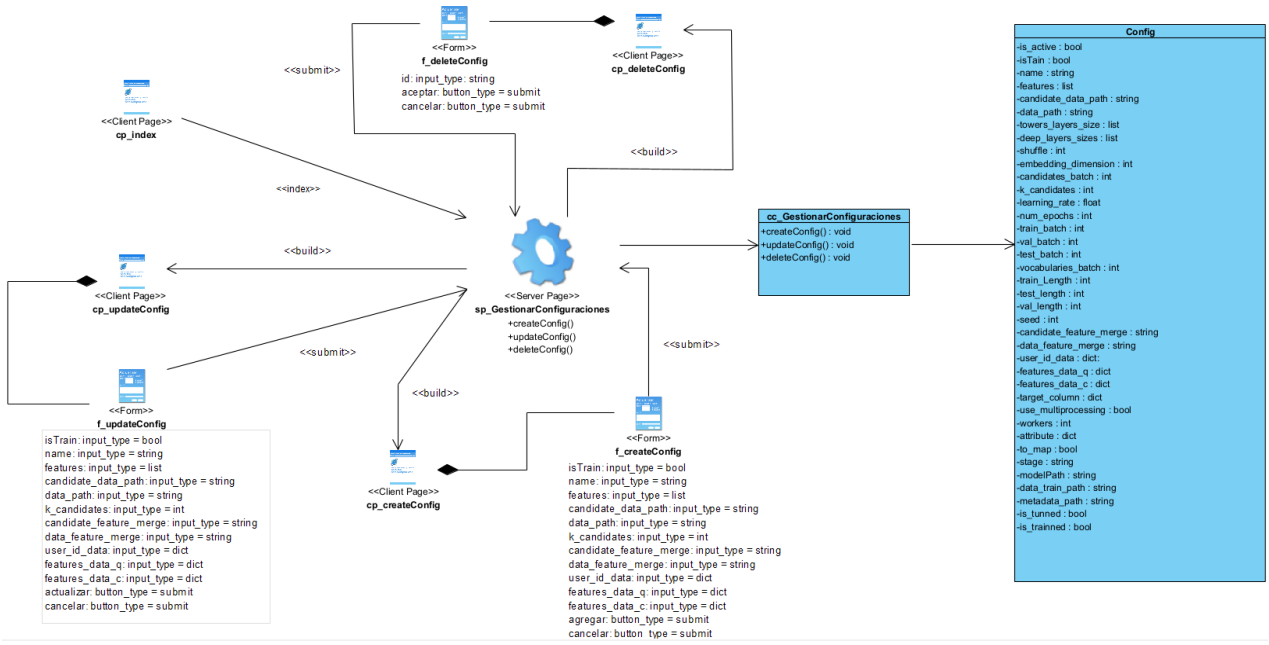


Figura 18: Diagrama de clases con estereotipos web (Fuente: Elaboración propia).

El diagrama de clases de la figura 18 se le aplico al caso de uso gestionar configuraciones. Se obtuvieron 7 diagramas de este tipo en total pero se dejó como muestra este en particular ya que es un requisito fundamental en el sistema debido a que todos los modelos utilizados trabajan en base a las configuraciones además de ser uno de los requisitos más grandes y delicados a desarrollar. Este

diagrama aporto un entendimiento general de cómo funciona el sistema y del funcionamiento de las configuraciones en particular.

2.5 Diagrama de secuencia.

Los diagramas de secuencia (DS) en el UML se usan principalmente para modelar las interacciones entre los actores y los objetos en un sistema, así como las interacciones entre los objetos en sí (González Matos, 2021; SOMMERVILLE, 2005).

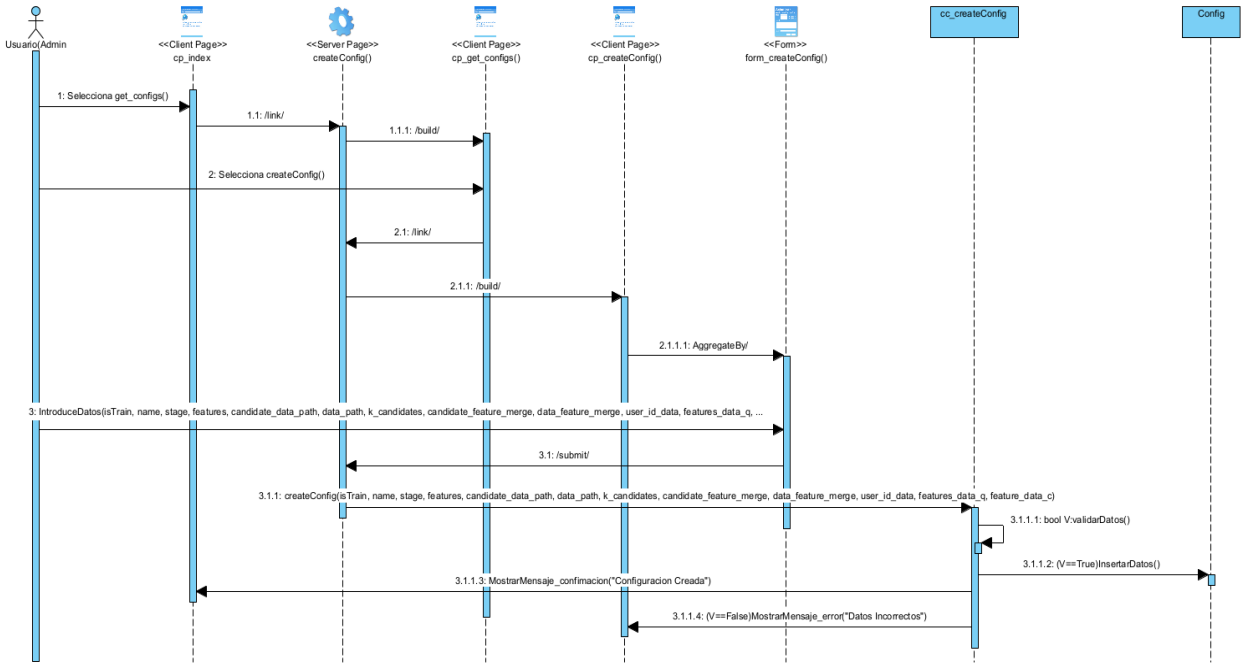


Figura 19: Diagrama de secuencia (Fuente: Elaboración propia).

El diagrama de secuencia de la figura 19 se le aplico al caso de uso gestionar configuraciones. El flujo representado en el diagrama de secuencia comienza cuando el actor (administrador) del sistema selecciona en el mismo la opción de añadir configuración. La página cliente intermediaria hace la solicitud a la server_page (página controladora/servidora). La página controladora construye la client_page (página cliente/vista) que permite añadir la configuración, esta hace link a la controladora que crea la página cliente con el formulario con los campos asociados a la configuración.

El actor introduce los datos que se envían a la server_page (página controladora/servidora) y automáticamente se adiciona la configuración en la clase controladora donde se validan que los datos estén correctos. En caso de

que los datos no hayan sido insertados de forma correcta se muestra un mensaje de error en la página cliente, en el caso contrario se muestra un mensaje de confirmación en la clase desde donde el actor realizo la solicitud inicial.

2.6 Modelo de datos.

Un modelo de base de datos muestra la estructura lógica del sistema, incluidas las relaciones y limitaciones que determinan cómo se almacenan los datos, la relación que existe entre sí, los procesos que los transforman y cómo se accede a ellos (González Matos, 2021; SOMMERVILLE, 2005).

Figura 20: Modelo de datos (Fuente: Elaboración propia).

Este diagrama (Figura 20) muestra las tablas con los datos que forman parte del sistema, los cuales se utilizan para que todas las funcionalidades del mismo puedan ponerse en marcha. No existen relaciones entre las tablas ya que se está utilizando una base de datos no relacional como es MongoDB (Ver Epígrafe 1.4.3). Este diagrama facilita la comprensión y el análisis de la lógica del sistema a partir de los datos que se manejan.

2.7 Requisitos de la propuesta de solución.

Según (Pressman, 2005), la tarea del análisis de requisitos es un proceso de descubrimiento, refinamiento, modelado, y especificación. Se refina en detalle el ámbito del software, y se crean modelos de los requisitos de datos, flujo de información y control, y del comportamiento operativo. El análisis de requisitos

permite al desarrollador especificar la función y el rendimiento del software, juntar la interfaz del software con otros elementos del sistema y establecer las restricciones que debe cumplir el software.

2.7.1 Requisitos Funcionales.

Los requisitos funcionales (RF) son declaraciones de las funcionalidades que debe cumplir el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Pressman, 2005).

Tabla 2: Descripción de los Requisitos Funcionales (Fuente: Elaboración propia).

No.	Requisito	Descripción	Prioridad	Complejidad
RF1.	Entrenar Modelos	El sistema debe poder entrenar los modelos configurados.	Alta	Alta
RF2.	Actualizar Modelos	El sistema debe poder actualizar los modelos entrenados previamente con nuevos datos.	Alta	Alta
RF3.	Recomendar	El sistema debe generar recomendaciones para las plataformas.	Alta	Alta
RF4.	Crear Configuraciones	El sistema debe ofrecer al administrador la opción de registrar configuraciones de los modelos.	Media	Media
RF5.	Actualizar Configuraciones	El sistema debe ofrecer al administrador la opción de editar configuraciones de los modelos.	Media	Media
RF6.	Mostrar Configuraciones	El sistema debe mostrar al administrador el detalle de la configuración que seleccione.	Baja	Baja
RF7.	Listar Configuraciones	El sistema debe listar al administrador las configuraciones disponibles.	Baja	Baja
RF8.	Crear Motor	El sistema debe ofrecer al administrador la opción de crear motores. Los cuales son instancias del SR.	Media	Media
RF9.	Actualizar Motor	El sistema debe ofrecer al administrador posibilidad de editar un motor ya creado.	Baja	Media

RF10.	Mostrar Motor	El sistema debe mostrar al administrador el detalle del motor que seleccione.	Media	Media
RF11.	Listar Motores	El sistema debe listar al administrador los motores disponibles.	Media	Baja
RF12.	Autenticar Administrador	El sistema debe permitir que los administradores inicien sesión en el sistema.	Alta	Media
RF13.	Registrar Administrador	El sistema debe permitir registrar un nuevo administrador.	Media	Baja

Los requisitos del sistema fueron recopilados mediante el método de entrevista, realizada al MSc. Yadier Perdomo Cuevas (Ver Anexo 1), uno de los tutores de esta investigación y miembro del equipo del proyecto Z17, encargado de las plataformas Picta, toDus y Apklis. Durante la entrevista, el tutor detalló todas las funcionalidades necesarias para el sistema en desarrollo.

La prioridad de cada requisito fue evaluada en tres categorías: baja, media y alta, con base en el grado de necesidad del requisito para que el sistema pueda cumplir sus funciones básicas y ser útil. Asimismo, la complejidad también fue clasificada en las mismas categorías, considerando el nivel de dificultad técnica requerido para implementar cada requisito funcional.

2.7.2 Requisitos no funcionales.

Los requisitos no funcionales (RnF) hacen referencia a las propiedades emergentes del sistema: fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento, son limitaciones sobre servicios o funciones que ofrece el mismo (González Matos, 2021; SOMMERVILLE, 2005).

Tabla 3: Descripción de los Requisitos no Funcionales (Fuente: Elaboración propia).

N.º	Descripción
Usabilidad	

RnF1	El sistema de recomendación debe estar contenido en una aplicación web.
RnF2	La aplicación debe presentar una interfaz agradable e intuitiva.
Seguridad	
RnF3	La información manejada por el sistema está protegida de acceso no autorizado de usuarios, definiéndose los permisos según sus roles.
RnF4	A los usuarios autorizados se les deberá garantizar el acceso a la información solicitada en todo momento.
Rendimiento	
RnF5	El sistema debe permitir que existan al menos 500 usuarios conectados de forma simultánea haciendo peticiones concurrentes.
RnF6	El tiempo de demora de una petición al servidor debe ser menor de tres (3) segundos aproximadamente.
Restricciones de Implementación y Diseño	
RnF7	El sistema debe ser desarrollado en su totalidad con tecnologías de código abierto.
RnF8	Servidor web FastAPI v0.115.0.
RnF9	Servidor de base de datos MongoDB v6.0.3
RnF10	Como lenguaje de programación se debe utilizar Python v3.11.9

Software	
RnF11	Para el uso del sistema se requiere una PC cliente con cualquier sistema operativo, que se puedan instalar navegadores webs para el uso de la aplicación.
Hardware	
RnF12	Se requiere un mínimo de 80gb de disco duro, una tarjeta de red de 100MB, un procesador Core i3-4170 a 3.70GHz y 16GB de memoria RAM. Se requieren estos requisitos mínimos para el entrenamiento de un modelo con un corpus de datos de 1 millón de filas.

Los requisitos funcionales y no funcionales desempeñaron un papel crucial en el proceso de desarrollo de la investigación permitiendo definir y analizar con exactitud todas las características y funciones deseadas para la propuesta de solución. Se establecieron todas las metas y objetivos necesarios para la implementación de este sistema.

2.8 Arquitectura de software.

El sistema entero está formado por dos partes: un núcleo, el cual tiene una estructura basada en la explicación dada en el epígrafe 2.1 donde se desarrolla la propuesta de solución. Existen diferentes capas de abstracción como:

- Las Etapas
- Los Modelos
- Las acciones

Donde esta última es la capa de comunicación o API (Application Programming Interface) (Ver Glosario de Términos página 101) del sistema con las cuales se

pueden realizar diferentes acciones como entrenar el sistema, usarlo, actualizarlo y post procesamiento de datos.

Este núcleo está envuelto por un servidor o backend el cual se encarga de crear las APIs que permiten al sistema comunicarse con otros y viceversa para realizar las integraciones. Este envoltorio backend se llevó a cabo por el Framerwork web FastAPI, desarrollado en Python, con el cual se utiliza la arquitectura Modelo-Vista-Controlador (MVC).

La idea detrás de MVC es que cada uno de los componentes en el código tenga un propósito, y que esos propósitos sean diferentes. Además de que la forma en que se relacionan estas partes ayuda con la ventaja de realizar un mejor mantenimiento en el futuro. Este patrón permite una separación muy clara de los datos de la aplicación que consta de tres partes interconectadas: vista, modelo y controlador (Sánchez, 2020).

- **Vista**: este elemento hace referencia a la parte de una aplicación que considera la interfaz gráfica. Es decir, cada elemento gráfico que interactúa con el usuario forma parte de la vista.
- **Modelo**: esta capa tiene la función de relacionar y gestionar los datos con los cuales la aplicación va a operar, como consultas, actualizaciones, creación de información o eliminación. Todo esto se le denomina como, Lógica de Negocio.
- **Controlador**: este componente responde ante eventos o acciones que realiza el usuario a través de la Vista para poder solicitar una operación de la información.

Al utilizar la arquitectura Modelo Vista Controlador (MVC) se separó la lógica del sistema de recomendación o el núcleo y la base de datos en la capa modelo, la vista está dada por una interfaz gráfica que permite manejar fácilmente la lógica del sistema. Como controlador se utiliza el Framerwork web FastAPI para conectar la lógica del sistema con la interfaz gráfica o con las plataformas que necesitan generar las recomendaciones. Este patrón arquitectónico ha permitido una mejor estructura de la aplicación facilitado la organización y separación de la misma.

2.9 Patrones de Diseño.

Los **patrones de diseño** son técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de lógica o interfaces. Un patrón de diseño es una solución comprobada y reutilizable para problemas de diseño, efectiva en resolver situaciones similares y aplicables a distintos problemas en diversas circunstancias (Pressman, 2005).

2.9.1 Patrones GRASP.

En diseño orientado a objetos, GRASP (General Responsibility Assignment Software Patterns) son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (Botero Tabares, 2023).

- **Experto.**

El patrón experto es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (Botero Tabares, 2023).

Este patrón ha sido utilizado en los modelos de aprendizaje (Ver Epígrafe 2.1.2), cada modelo se encapsuló como una clase diferente guardando cada uno sus propias tareas y acciones (Ver Figura 21).

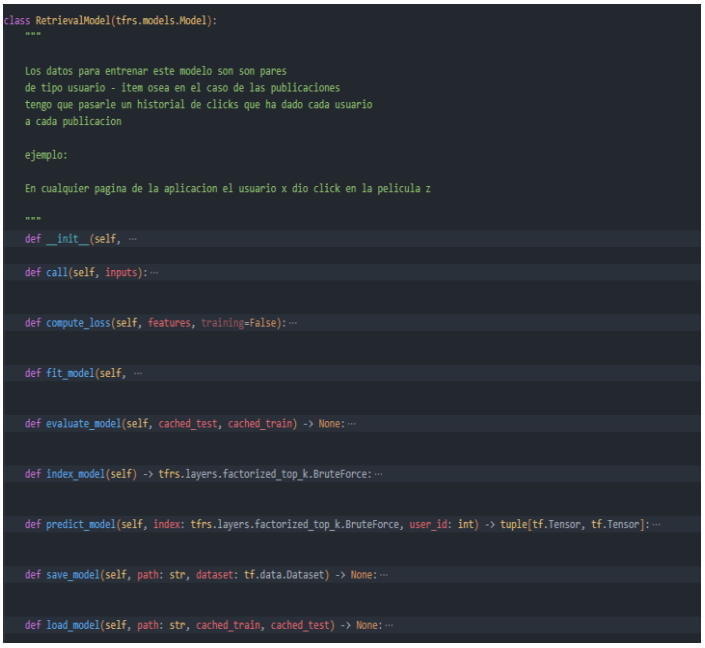


Figura 21: Clase "RetrievalModel" la cual representa el modelo de recuperación donde se refleja el patrón Experto (Fuente: Elaboración propia).

- **Creador.**

El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciado) de nuevos objetos o clases. Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización (Botero Tabares, 2023).

Para implementar este patrón se han utilizado "acciones", estas "acciones" no son más que funciones que permiten ejecutar el sistema. Estas funciones son train (permite entrenar el sistema), fine_tunning (permite actualizar el sistema ya entrenado con nuevos datos) y use_engine (permite generar recomendaciones). Estas acciones son las encargadas de implementar el patrón **creador** instanciando modelos y clases para el manejo de datos (Ver Figura 22).

```python
def train():
    global engine

    # General Configs
    engine_name = "Engine_v0.2"
    service_models_path = f"service_models/{engine_name}"
    delete_path(service_models_path)

    # Retrieval Configs
    retrieval_config = ModelConfig(
        model_name="Retrieval_lite",
        features=['usuario_id', 'id'],
        data_paths=[
            "../../datasets/picta_publicaciones_procesadas_sin_nulas_v2.csv",
            "../../datasets/vistas_no_nulas.csv"
        ],
        towers_layers_sizes=[],
        shuffle=10_000,
        embedding_dimension=64,
        candidates_batch=128,
        k_candidates=100,
        learning_rate=0.1,
        num_epochs=1,
        use_multiprocessing=True,
        workers=4,
        train_batch=8192,
        val_batch=4096,
        test_batch=4096,
        vocabularies_batch=1000,
        train_length=60,
        test_length=20,
        val_length=20,
        seed=42,
        features_data_q={
            'usuario_id': { 'dtype': FeaturesTypes.CategoricalInteger, 'w': 1 },
            # 'timestamp': { 'dtype': CategoricalContinuous.CategoricalContinuous, 'w': 0.3 }
        },
        features_data_c={
            'id': { 'dtype': FeaturesTypes.CategoricalInteger, 'w': 1 },
            # 'nombre': { 'dtype': StringText.StringText, 'w': 0.2 },
            # 'descripcion': { 'dtype': StringText.StringText, 'w': 0.1 }
        }
    )

    # Likes Configs
    likes_config = ModelConfig(
        model_name="Likes_lite",
        features=['usuario_id', 'id', "like_dislike"],
        data_paths=["../../datasets/likes.csv"],
        towers_layers_sizes=[],
        deep_layers_sizes = [],
        shuffle=10_000,
        embedding_dimension=64,
        learning_rate=0.0001,
        num_epochs=1,
        use_multiprocessing=True,
        target_column={
            "current": "valor",
            "new": "like_dislike"
        },
        workers=4,
        train_batch=8192,
        val_batch=4096,
        test_batch=4096,
        vocabularies_batch=1000,
        train_length=60,
        test_length=20,
        val_length=20,
        seed=42,
        features_data_q={
            'usuario_id': { 'dtype': FeaturesTypes.CategoricalInteger, 'w': 1 },
            # 'timestamp': { 'dtype': CategoricalContinuous.CategoricalContinuous, 'w': 0.3 }
        },
        features_data_c={
            'id': { 'dtype': FeaturesTypes.CategoricalInteger, 'w': 1 },
            # 'nombre': { 'dtype': StringText.StringText, 'w': 0.2 },
            # 'descripcion': { 'dtype': StringText.StringText, 'w': 0.1 }
        }
    )


    print(f"***** Proceso de Entrenamiendo del Systema {engine_name} Iniciado *****")

    pipe = DataPipeline()
    pubs_df, views_df = pipe.read_csv_data(paths=retrieval_config.data_paths)
    pubs_df = pubs_df[: 5000]

    views_df = views_df[: retrieval_config.shuffle]
    views_df = views_df.drop(['id'], axis=1)
    pubs_df['descripcion'] = pubs_df['descripcion'].astype(str)
    pubs_df['nombre'] = pubs_df['nombre'].astype(str)


    views_df = pipe.merge_data(
        left_data=views_df,
        right_data=pubs_df,
        left_on="publicacion_id",
        right_on="id",
        output_features=retrieval_config.features
    )

    pubs_ds = pipe.convert_to_tf_dataset(pubs_df)
    views_ds = pipe.convert_to_tf_dataset(views_df)

    vocabularies = pipe.build_vocabularies(
        features=retrieval_config.features,
        ds=views_ds,
        batch=retrieval_config.vocabularies_batch
    )

    total, train_length, val_length, test_length = pipe.get_lengths(
        ds=views_ds,
        train_length=retrieval_config.train_length,
        test_length=retrieval_config.test_length,
        val_length=retrieval_config.val_length
    )

    train, val, test = pipe.split_into_train_and_test(
        ds=views_ds,
        shuffle=retrieval_config.shuffle,
        train_length=train_length,
        val_length=val_length,
        test_length=test_length,
        seed=retrieval_config.seed
    )

    cached_train, cached_val, cached_test = pipe.data_caching(
        train=train,
        val=val,
        test=test,
        shuffle=retrieval_config.shuffle,
        train_batch=retrieval_config.train_batch,
        val_batch=retrieval_config.val_batch,
        test_batch=retrieval_config.test_batch
    )

    pipe.close()

    retrieval_model = retrieval_stage.retrieval_model(
        # model_name=retrieval_model_name,
        # towers_layers_sizes=towers_layers_sizes,
        vocabularies=vocabularies,
        # features_data_q=features_data_q,
        # features_data_c=features_data_c,
        # embedding_dimension=embedding_dimension,
        # test=test,
        # shuffle=10_000,
        # test_batch=512,
        candidates=pubs_ds,
        # candidates_batch=candidates_batch,
        # k_candidates=k_candidates
        config=retrieval_config
    )

    retrieval_model.fit_model(
        cached_train=cached_train,
        cached_val=cached_val,
        # learning_rate=learning_rate,
        # num_epochs=num_epochs,
        # use_multiprocessing=use_multiprocessing,
        # workers=workers
    )
    retrieval_model.evaluate_model(
        cached_test=cached_test,
        cached_train=cached_train
    )
    retrieval_model.save_model(
        service_models_path, views_ds)

    pipe = DataPipeline()
    likes_df, = pipe.read_csv_data(paths=[
        "../../datasets/likes.csv"
    ])

    likes_df = likes_df[: likes_config.shuffle]
    likes_df = likes_df.drop(['id'], axis=1)
    likes_df[likes_config.target_column['new']] = likes_df[likes_config.target_column['current']]\
        .map({True: 1, False: 0})

    likes_df = pipe.merge_data(
```

Figura 22: Acción Entrenar encargada de crear instancias de modelos (Fuente: Elaboración propia).

- **Controlador.**

Controlador es un patrón de diseño estructural que establece una clase para recibir y manejar eventos o solicitudes iniciadas por la interfaz de usuario, actuando como intermediario entre la capa de presentación (vista) y la capa de lógica de negocio (modelo) (Botero Tabares, 2023).

```python
# Create a new engine
@router.post("/engines")
async def create_engine(engine: EngineUserInput):
    try:
        existing_engine = engine_collection.find_one({"name": engine.name})
        if existing_engine:
            raise HTTPException(
                status_code=400,
                detail="El nombre del engine ya existe"
            )

        if engine.is_active:
            active_engine = engine_collection.find_one({"is_active": True})
            if active_engine:
                raise HTTPException(status_code=400, detail="Ya existe un engine activo")

        engine_data = jsonable_encoder(engine)
        engine_data['createAt'] = datetime.now()
        engine_data['service_models_path'] = ''
        new_engine = engine_collection.insert_one(engine_data)
        return { "id": str(new_engine.inserted_id) }
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```

Figura 23: Función controladora de la funcionalidad: “crear motor” donde se refleja el patrón controlador (Fuente: Elaboración propia).

Este patrón ha sido utilizado en la capa “controlador” del sistema donde están todas las APIs (Ver Glosario de Términos página 101) que conectan la lógica de la aplicación con la vista o interfaz. Este permite una separación clara de responsabilidades, facilitando la gestión y el mantenimiento del código, al tiempo que mejora la escalabilidad del sistema.

- **Alta cohesión y bajo acoplamiento.**

El patrón **Alta cohesión** significa que un módulo se enfoca en una tarea específica y todas sus funciones están relacionadas, lo que mejora la claridad, mantenibilidad y reutilización del código. **Bajo acoplamiento** implica que los módulos tienen pocas dependencias entre sí, permitiendo que los cambios en uno afecten mínimamente a otros, facilitando la escalabilidad y el testing (Botero Tabares, 2023).

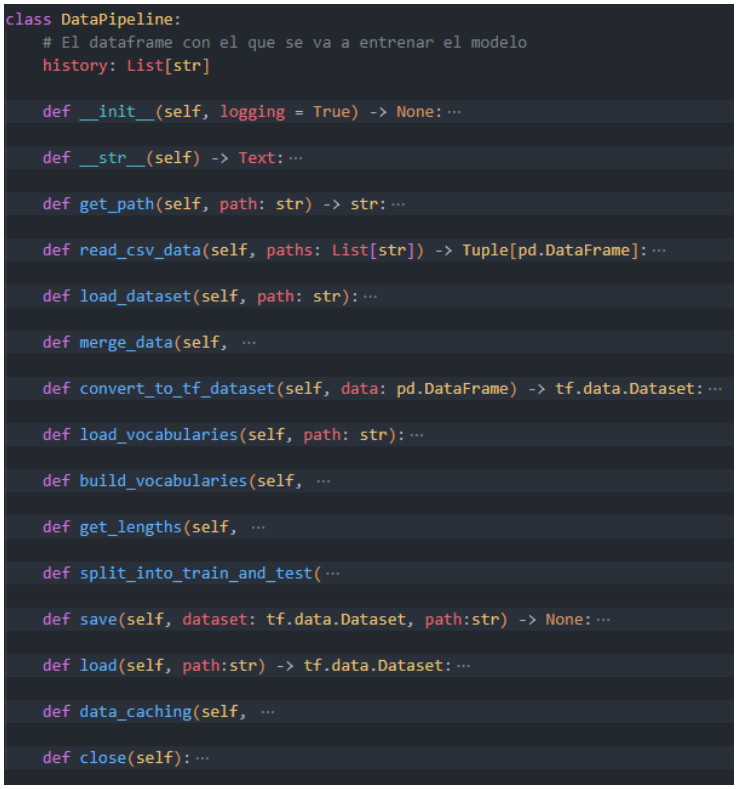


Figura 24: Clase DataPipeline donde se refleja el patrón: Alta Cohesión (Fuente: Elaboración Propia).

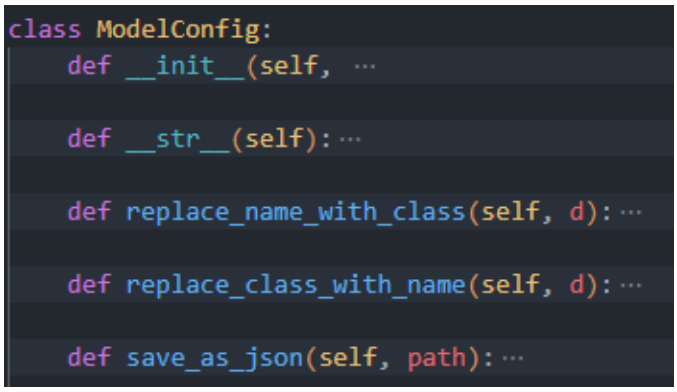


Figura 25: Clase ModelConfig la cual se utiliza para poner en práctica el patrón: Bajo Acoplamiento (Fuente: Elaboración Propia).

La clase DataPipeline (Figura 24) muestra una alta cohesión, ya que todas sus responsabilidades están estrechamente relacionadas con el procesamiento y manipulación de datos. La clase ModelConfig (Figura 25) es utilizada para pasar configuraciones a los modelos lo que reduce el acoplamiento entre la configuración y la implementación del modelo. Estos dos patrones permiten crear sistemas más flexibles, fáciles de mantener y robustos, ideales para proyectos complejos como

un sistema de recomendación basado en Deep Learning como el de esta investigación.

2.9.2 Patrones GOF.

- **Singleton.**

Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (Ver Figura 26) (Guerrero et al., 2013). Este patrón fue utilizado para tener una instancia global accesible desde cualquier parte de la aplicación de las etapas del sistema.

```python
class RetrievalStage(AbstractStage):
    _instance = None  # Atributo de clase para almacenar la instancia única
    model: RetrievalModel
    name: str

    def __new__(cls, *args, **kwargs):
        if cls._instance is None:
            cls._instance = super(RetrievalStage, cls).__new__(cls)
        return cls._instance

    def __init__(self) -> None:
        if not hasattr(self, 'initialized'):  # Verifica si ya ha sido inicializado
            self.model = RetrievalModel
            self.name = "retrieval"
            self.initialized = True  # Marca como inicializado
```

Figura 26: Clase RetrievalStage donde se pone de manifiesto el patrón Singleton (Fuente: Elaboración Propia).

```python
class RankingStage(AbstractStage):
    _instance = None  # Atributo de clase para almacenar la instancia única
    model: LikesModel
    name: str

    def __new__(cls, *args, **kwargs):
        if cls._instance is None:
            cls._instance = super(RankingStage, cls).__new__(cls)
        return cls._instance

    def __init__(self) -> None:
        if not hasattr(self, 'initialized'):  # Verifica si ya ha sido inicializado
            self.model = LikesModel
            self.name = "ranking"
            self.initialized = True  # Marca como inicializado
```

Figura 27: Clase RankingStage donde se pone de manifiesto el patrón Singleton (Fuente: Elaboración Propia).

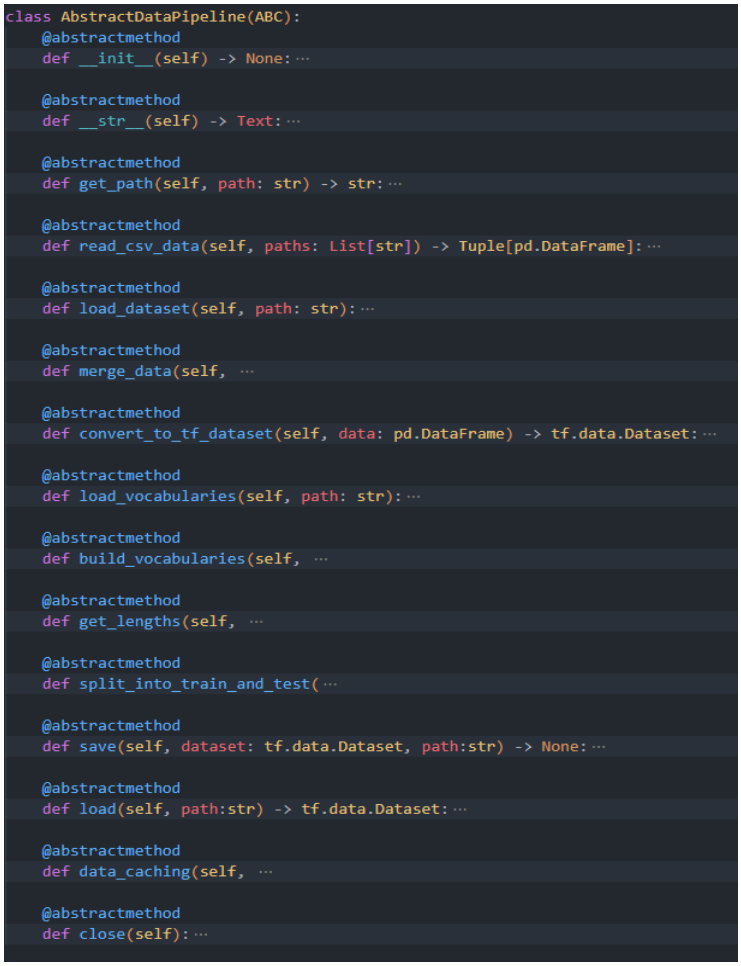
```
from stages.RetrievalStage import RetrievalStage
from stages.RankingStage import RankingStage

retrieval_stage = RetrievalStage()
ranking_Stage = RankingStage()
```

Figura 28: Creación de dos instancias globales en el sistema (Fuente: Elaboración propia).

En las clases **RetrievalStage** (Figura 26) y **RankingStage (**Figura 27) se implementa la lógica que asegura que cada función tenga una única instancia en toda la aplicación, sin importar cuántas veces se intente crear. Como se ilustra en la Figura 28, se generan dos instancias que permiten acceder a estas clases desde cualquier parte de la aplicación. Esto permite una gestión centralizada del estado y la configuración, facilitando la coordinación entre componentes y asegurando la consistencia de los datos en todo el sistema.

- **Template Method.**

Define en una operación el esqueleto de una clase, delegando en las subclases algunos de sus pasos. Esto permite que las subclases redefinir ciertos pasos de un algoritmo sin cambiar su estructura (Guerrero et al., 2013).



```python
class AbstractDataPipeline(ABC):
    @abstractmethod
    def __init__(self) -> None:...

    @abstractmethod
    def __str__(self) -> Text:...

    @abstractmethod
    def get_path(self, path: str) -> str:...

    @abstractmethod
    def read_csv_data(self, paths: List[str]) -> Tuple[pd.DataFrame]:...

    @abstractmethod
    def load_dataset(self, path: str):...

    @abstractmethod
    def merge_data(self, ...

    @abstractmethod
    def convert_to_tf_dataset(self, data: pd.DataFrame) -> tf.data.Dataset:...

    @abstractmethod
    def load_vocabularies(self, path: str):...

    @abstractmethod
    def build_vocabularies(self, ...

    @abstractmethod
    def get_lengths(self, ...

    @abstractmethod
    def split_into_train_and_test(...

    @abstractmethod
    def save(self, dataset: tf.data.Dataset, path:str) -> None:...

    @abstractmethod
    def load(self, path:str) -> tf.data.Dataset:...

    @abstractmethod
    def data_caching(self, ...

    @abstractmethod
    def close(self):...
```

Figura 29: Clase interfaz que se utiliza como plantilla para la clase DataPipeline (Fuente: Elaboración propia).

La clase AbstractDataPipeline en la figura 29 es una plantilla de la clase DataPipeline que contiene todos los métodos, atributos, entrada, salida y tipos lo cual facilita la reutilización de código al centralizar la estructura general de un algoritmo en una clase base, permitiendo que las subclases personalicen detalles específicos. Esto promueve consistencia, flexibilidad y extensibilidad, asegurando que los pasos claves sigan un flujo definido.

2.9.3 Conclusiones de los patrones utilizados.

El uso de estos patrones en el desarrollo de la solución fue fundamentales para lograr un sistema robusto y mantenible. El patrón Experto permitió delegar responsabilidades de forma adecuada, mejorando la cohesión; mientras que el Creador garantizó una construcción consistente de objetos complejos. El Controlador organizó eficazmente la interacción entre el usuario y el sistema,

separando la lógica de negocio de la interfaz. Además, la aplicación de principios como alta cohesión y bajo acoplamiento fortaleció la modularidad y flexibilidad del sistema.

El Singleton aseguró la existencia de instancias únicas para componentes globales, optimizando recursos y datos compartidos. Por último, el Template Method facilitó la reutilización del código al definir estructuras comunes y permitir su personalización. Estos patrones, en conjunto, aseguraron una arquitectura modular, escalable y fácil de mantener, alineada con las mejores prácticas de diseño de software.

2.10 Conclusiones del capítulo.

Como parte del desarrollo del presente capítulo se determinan las siguientes conclusiones parciales:

- ✓ El análisis de las características del sistema y la modelación del dominio permitió identificar los principales requisitos funcionales y no funcionales del sistema de recomendaciones, los cuales fueron agrupados y categorizados por casos de uso.

- ✓ El diseño de los diagramas de clases permitió el entendimiento sobre la composición física y lógica del sistema.

- ✓ Los artefactos generados según la metodología de desarrollo utilizada y los patrones de arquitectura y diseño descritos, constituyeron una guía fundamental para la construcción de la propuesta de solución.

CAPÍTULO 3: Validación y prueba del Sistema de Recomendación basado en Deep Learning para las plataformas del proyecto Z17.

El presente capítulo está orientado a las pruebas y validación de las variables de la investigación que se llevan a cabo para dar total cumplimiento a los requisitos establecidos por el cliente. Se evidencian las pruebas aplicadas a la solución con el objetivo de comprobar las funcionalidades en los diferentes escenarios y así verificar en todos los casos que los resultados sean los esperados.

3.1 Modelo de despliegue.

Un modelo de despliegue es un diagrama estructurado que muestra la arquitectura del sistema desde el punto de vista de distribución de los artefactos del software en los destinos de despliegue; se definen los artefactos como representaciones de elementos concretos en el mundo físico que son el resultado de un proceso de desarrollo (González Matos, 2021).

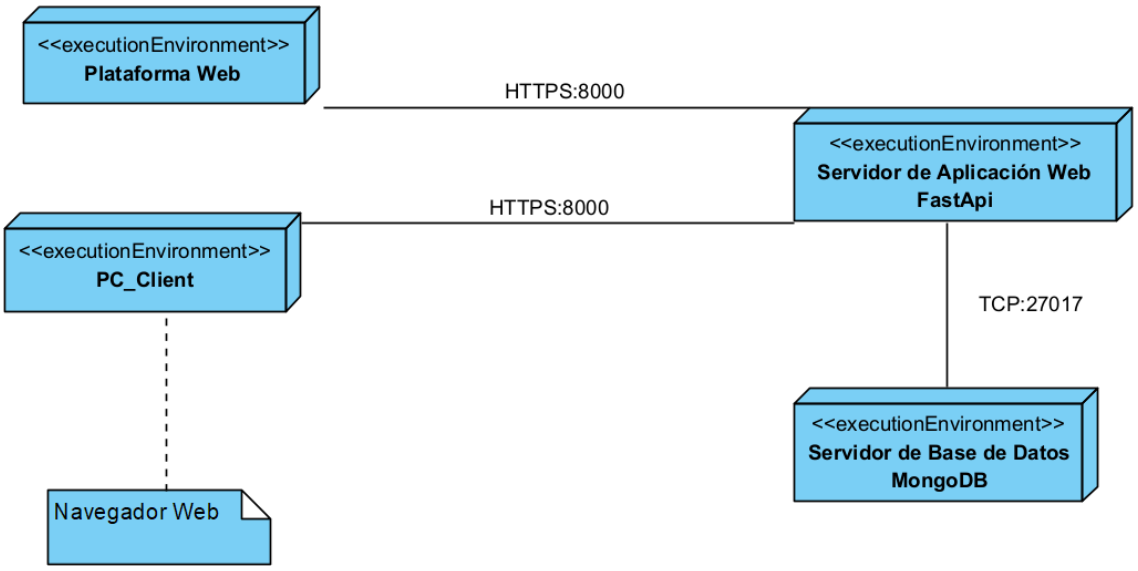


Figura 30: Diagrama de despliegue (Fuente: Elaboración propia).

El diagrama de despliegue es una herramienta clave en el desarrollo de este sistema, ya que permite visualizar cómo los componentes del sistema se distribuyen físicamente en la infraestructura tecnológica. Su realización aporta claridad en la arquitectura física, optimiza recursos, mejora la comunicación

entre equipos, previene errores y facilita la planificación de escalabilidad y mantenimiento. Además, asegura la alineación con los requisitos del negocio, minimiza riesgos de fallos y sirve como documentación futura. En esencia, garantiza una implementación más eficiente, segura y adaptable del sistema.

3.1.1 Descripción de elementos e interfaces de comunicación.

Dispositivo PC_Cliente: La estación de trabajo necesita un navegador web para conectarse al sistema hospedado en el servidor de aplicaciones utilizando el protocolo de comunicación HTTPS y el puerto 8000. Se utiliza el puerto 8000 ya que este es el puerto que utiliza el servidor web de FastAPI por defecto.

Servidor de aplicación web: Es la estación de trabajo que hospeda el código fuente de la aplicación y que le brinda al usuario las interfaces para realizar los procesos del sistema. Esta estación se comunica con el servidor de base de datos donde se almacenan los datos de la aplicación realizando la comunicación mediante el protocolo TCP (Transmission Control Protocol).

Servidor de Base de Datos (MongoDB): Este servidor es el encargado del almacenamiento de los datos del sistema. Se comunica con el servidor de aplicaciones del sistema mediante el protocolo TCP y el puerto 27017, posibilitando el acceso mediante el usuario con privilegios para las operaciones determinadas a realizarse en el mismo.

Plataformas: Son las plataformas del proyecto Z17 (Apklis, Picta y toDus) las cuales van a consumir del servidor web para obtener las recomendaciones que necesitan mostrar a los usuarios utilizando el protocolo de comunicación HTTPS y el puerto 8000.

3.2 Estándares de codificación.

Con vistas a mejorar el entendimiento del código perteneciente al sistema de recomendación por otros desarrolladores y alcanzar una uniformidad en el mismo, a continuación se muestran algunos estándares de codificación utilizados en la implementación.

- **Indentación**: Usar 4 espacios por nivel de indentación en cada linea.

- **Nombres de variables y funciones**:
    - Variables y funciones: snake_case (se utilizan letras minúsculas y separando las palabras con guiones bajos).
    - Clases: PascalCase (cada palabra comienza con una letra mayúscula y no se utilizan guiones bajos).
    - Constantes: UPPER_CASE (se utilizan letras mayúsculas y separando las palabras con guiones bajos).
- **Espaciado**: Evitar espacios innecesarios alrededor de operadores, llaves y comas.
- **Tipo de datos en funciones**: Usar anotaciones de tipo para todos los parámetros y valores de retorno de las funciones, clases y atributos.
- **Modularización del Código**: Dividir el código en funciones o clases separadas para cada acción o función que realiza el sistema de recomendaciones (actualización, entrenamiento y generación).

Definir los estándares de codificación para este proyecto ha aportado importantes beneficios, como una mayor coherencia en el desarrollo, lo que facilita la colaboración y mejora la legibilidad del código. Esto ha garantizado el mantenimiento, permitiendo que los desarrolladores comprendan rápidamente el código, reduciendo tiempos de búsqueda y corrección de errores. Además, ha favorecido la escalabilidad del sistema, facilitando la integración de nuevas funcionalidades. También ha mejorado la gestión de errores y la seguridad, minimizando riesgos y fallos operativos. En general, estos estándares han aumentado la eficiencia del equipo y la robustez del sistema.

3.3 Estrategia de prueba.

Para garantizar el completo funcionamiento de toda aplicación informática es necesario realizar pruebas para su validación y de esta forma evitar cualquier problema o insatisfacción por parte de los clientes. Estas pruebas se trazan mediante estrategias que permitan evaluar todos los aspectos de determinado producto teniendo en cuenta su característica y así de una forma u otra poder garantizar el éxito del mismo (SOMMERVILLE, 2005).

Antes de continuar es necesario aclarar un punto importante en las pruebas de esta solución. Es necesario recalcar que solo se utilizó a la Plataforma Picta para las presentes pruebas en este capítulo ya que por problemas de tiempo no alcanza para probar el sistema con las tres plataformas ya que cada una lleva sus propias configuraciones, adaptaciones y un entrenamiento dedicado. Esto es un proceso lento, muy delicado y requiere de tiempo y muchas iteraciones de prueba para cada aplicación. Por tanto se ha decidido aplicar las pruebas únicamente a la plataforma Picta pero dejar marcado que la propuesta de solución de esta investigación está lista para ser utilizada con las tres plataformas sin ningún problema.

3.3.1 Datos de entrenamiento.

Para entrenar el sistema de recomendación se necesitan datos, datos que permitan al modelo aprender sobre las preferencias de los usuarios para así recomendar elementos personalizados e interesantes. A continuación se mostrará una tabla con los datos utilizados y algunos rasgos de estos datos. La tabla está formada por 4 columnas que describen los datos utilizados, a continuación una descripción de estas columnas:

- **Datasets:** los nombres de las colecciones de los datos utilizados.
- **Descripción:** una breve descripción de los datos utilizados.
- **Características:** se le llama característica a las variables o atributos que describen los datos y que el modelo utiliza para hacer predicciones o clasificaciones. Este campo dice cuáles son las características utilizadas en cada dataset.
- **Cantidad:** indica el tamaño del dataset o cuantos datos hay en el conjunto de datos seleccionado y se nombra por filas.

Tabla 4: Descripción de los datos utilizados para entrenar los modelos (Fuente: Elaboración Propia).

Dataset	Descripción	Características	Cantidad

Publicaciones	El contenido audiovisual que se consume en la plataforma Picta se denomina Publicaciones, estas, están categorizada por tipologías como Película, Documental, Serie, entre otras.	Id, nombre, descripción y categoría.	22 882 filas
Vistas	Respecto a la retroalimentación implícita (Ver Epígrafe 1.1.2) se utilizaron las vistas o lo que es lo mismo los clicks de los usuarios en cada publicación.	usuario_id y fecha.	1 000 000 filas
Likes	Como retroalimentación explicita (Ver Epígrafe 1.1.2) se tomaron los likes y los dislikes, lo cual indica con precisión que publicaciones le gusta o no a cada usuario.	usuario_id, fecha y valor.	154 396 filas
Usuarios	Este dataset se utiliza para tomar características de los usuarios para que los modelos tengan mejor contexto y así dar mejor personalización a las recomendaciones.	usuario_id y fecha_nacimiento	783 484 filas

En el dataset de likes, había inicialmente 524 396 filas pero, tenía un problema llamado desbalanceo de clases o class imbalance. Esto ocurre cuando una clase está representada con muchos más ejemplos que la otra lo que provoca que un modelo de Deep Learning como los utilizados clasifique casi todos los elementos como la clase mayoritaria (Ghosh et al., 2024). En este caso la clase que más abundaba era la 0 en un 86% que representa a los dislikes, esto provocaría que el modelo de clasificación casi siempre evalúe como negativo cada candidato.

Para solventar este problema se ha utilizado la técnica de submuestreo o undersampling la cual consiste en reducir el número de ejemplos de la clase mayoritaria, eliminando algunos de sus datos para equilibrar la proporción entre las clases (Ghosh et al., 2024). En este caso en particular se eliminaron dislikes en base a la fecha de creados, quitando los más antiguos y así se logró balancear el dataset quedando un 49% de likes y un 51% de dislikes lo cual es un buen balance para entrenar un modelo de clasificación binaria de Deep Learning.

3.3.2 Configuraciones del sistema.

En este sub epígrafe se establecerán las configuraciones utilizadas para el entrenamiento de cada modelo (Ver Modelos en el Epígrafe 2.1.2). A continuación una descripción de cada configuración utilizada (Gao et al., 2023):

- **Learning Rate:** Representa la tasa de aprendizaje del modelo, es decir, cuánto se ajustan los pesos en cada paso de optimización.
- **Number of Epochs:** Indica cuántas veces el modelo recorrerá el conjunto completo de datos de entrenamiento.
- **Train Length** y **Test Length:** Reflejan la proporción del conjunto de datos asignada al entrenamiento y prueba, respectivamente.
- **Train Batch** y **Test Batch:** Especifican el tamaño del lote de datos procesado en cada paso durante el entrenamiento y la evaluación. Es decir al entrenar un modelo no se le pasa el conjunto de datos de entrenamiento entero sino que estos datos son divididos en pequeños lotes pasando uno a uno por el modelo.

Tabla 5: Configuraciones utilizadas para el proceso de entrenamiento de cada modelo (Fuente: Elaboración propia).

Modelo	Learning Rate	Number of Epochs	Train Length	Test Length	Train Batch	Test Batch
Recuperación	0.1	8	60%	40%	32	16
Clasificación	0.0001	8	60%	40%	512	256

Estas configuraciones son utilizadas para el proceso de entrenamiento y actualización de los modelos. Además, influyen directamente en el rendimiento del sistema (Ver Epígrafe 3.4.3) y su precisión, por lo que es importante encontrar un balance positivo entre ellas.

3.4 Pruebas realizadas al sistema.

Una vez finalizada la implementación del producto que se solicita, es necesario realizarle pruebas, con el objetivo de detectar errores en la aplicación y la documentación e identificar posibles fallos de implementación, calidad, o usabilidad del software. Además, medir el grado en que el software cumple con los requerimientos (Pressman, 2005).

3.4.1 Prueba de Caja Negra.

Según (Pressman, 2005), Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. Los casos de prueba de la caja negra pretenden demostrar que:

1. Las funciones del software son operativas.
2. La entrada se acepta de forma correcta.
3. Se produce una salida correcta.
4. La integridad de la información externa se mantiene.

Resultados de la prueba de caja negra.

La siguiente grafica muestra los resultados obtenidos en las pruebas realizadas al sistema durante cada iteración; en búsqueda de no conformidades de

funcionalidad, redacción e interfaz. Obteniendo un resultado satisfactorio durante la última iteración.

Tabla 6: Cantidad de no conformidades por cada iteración de las pruebas funcionales (Fuente: Elaboración propia).

No conformidades	1er iteración	2da iteración	3ra iteración	4ta iteración
Detectadas	8	3	1	0
Resueltas	4	5	3	0
Pendientes	4	2	0	0

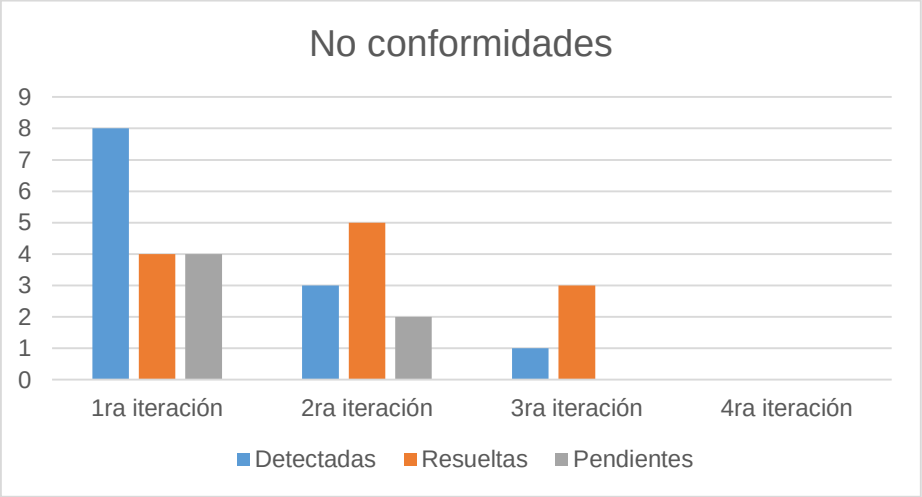


Figura 31: Comportamiento de las no conformidades de las pruebas funcionales ejecutadas (Fuente: Elaboración propia).

Como se muestra en la Figura 31 para la validación de los requisitos funcionales se realizaron tres iteraciones.

Iteración 1: Fueron detectadas 8 no conformidades; 2 de ortografía, 2 de interfaz y 4 de funcionalidad. De estas fueron resueltas 4; 2 de ortografía y 2 de interfaz quedando 4 no conformidades de funcionalidad pendiente.

Iteración 2: Fueron detectadas 3 no conformidades de interfaz. Se resolvieron en esta iteración un total de 5 no conformidades; 3 de interfaz y 2 de funcionalidad quedando pendiente 2 no conformidades de funcionalidad.

Iteración 3: Se detectó 1 no conformidad de funcionalidad. Se resolvieron 3 no conformidades de funcionalidad.

Iteración 4: No fue detectada ninguna no conformidad, por lo cual se puede decir que al final de las iteraciones quedan resueltas todas las no conformidades.

Diseño de los casos de prueba.

Tabla 7: Caso de prueba Entrenar (Fuente: Elaboración propia).

Caso de prueba: Entrenar
Nombre de requisito: Entrenar
Nombre de la persona que realiza la prueba: Alejandro Figueroa Rodríguez.
Descripción de la prueba: Prueba a la funcionalidad Entrenar. El objetivo de esta prueba es entrenar el sistema de recomendaciones en su totalidad.
Entrada/Pasos de ejecución: Se selecciona el motor que se quiere entrenar y se presiona el botón entrenar.
Evaluación de la prueba: Satisfactoria. El motor se entrena correctamente sin ningún error.

Tabla 8: Caso de prueba Generar Recomendaciones (Fuente: Elaboración propia).

Caso de prueba: Generar Recomendaciones
Nombre de requisito: Recomendar.
Nombre de la persona que realiza la prueba: Alejandro Figueroa Rodríguez.

Descripción de la prueba: Prueba a la funcionalidad Generar Recomendaciones.
Entrada/Pasos de ejecución: Se realizan las siguientes acciones: - Escoger un usuario. - Verificar sus preferencias. - Escoger una colección de documentos que se ajuste a sus preferencias. - Insertar datos de contexto.
Datos de contexto: • **usuario_id:** 424170. • **fecha:** 1732115445 = 20 de noviembre de 2024**.** • **fecha_nacimiento:** 999930930 = 29 de agosto de 2001.
Ejecutar la prueba.
Evaluación de la prueba: Satisfactoria. La funcionalidad Recomendar genera recomendación sin ningún error encontrado.

Tabla 9: Caso de prueba Actualizar (Fuente: Elaboración propia).

Caso de prueba: Actualizar
Nombre de requisito: Actualizar
Nombre de la persona que realiza la prueba: Alejandro Figueroa Rodríguez.
Descripción de la prueba: Prueba a la funcionalidad Actualizar. El objetivo de esta prueba es actualizar el sistema de recomendación con nuevos datos aparte de los que fueron utilizados.

Entrada/Pasos de ejecución: Se selecciona el motor que se quiere actualizar y se presiona el botón actualizar.
Datos: • 1000 Vistas. • 1000 Likes y Dislikes.
Evaluación de la prueba: Satisfactoria. El motor se actualiza correctamente con los datos de prueba sin lanzar ningún error.

3.4.2 Prueba de Caja Blanca.

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo (Pressman, 2005).

Técnica del camino básico.

Es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Iglesias Mizrahi, 2016) (Pressman, 2005).

Complejidad Ciclomática.

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Cuando se usa en el contexto del método de prueba del camino básico, el valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de

pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez (Iglesias Mizrahi, 2016).

```python
@router.post("/signin", response_class=HTMLResponse)
async def signin(
    username: str = Form(...),
    password: str = Form(...)
):
    from main import SECRET_KEY
    try:
        collection = db["users"]

        # Buscar el usuario por email
        user = collection.find_one({"username": username})

        if not user or not pwd_context.verify(password, user["password"]):
            raise HTTPException(
                status_code=status.HTTP_401_UNAUTHORIZED,
                detail="Credenciales incorrectas"
            )

        token_data = {
            "sub": username,
            "exp": datetime.utcnow() + timedelta(days=30)  # El token expira en 30 días
        }
        token = jwt.encode(token_data, SECRET_KEY, algorithm="HS256")

        # Crear una respuesta de redirección
        response = RedirectResponse(
            "/config",
            status_code=status.HTTP_303_SEE_OTHER
        )

        # Establecer la cookie con el token
        response.set_cookie(
            key="auth_token",
            value=token,
            httponly=True
        )

        return response
    except HTTPException as http_exc:
        raise http_exc
    except Exception as e:
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=str(e)
        )
```

Figura 32: Función controladora para autenticar usuario (Fuente: Elaboración propia).

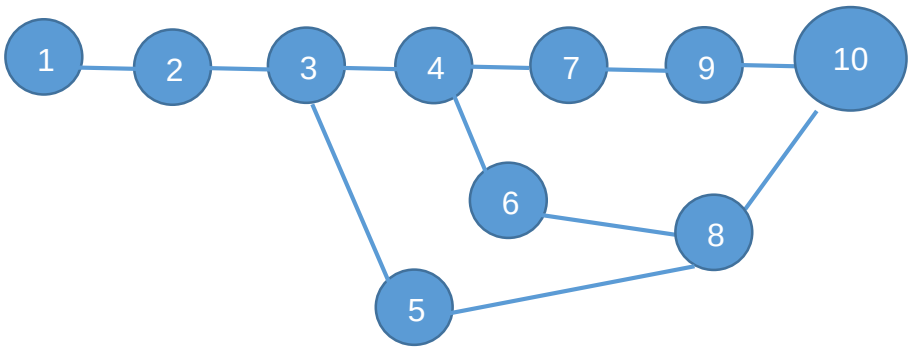


Figura 33: Técnica del camino básico (Fuente: Elaboración propia).

Cantidad de caminos mínimos = 3

Camino 1: 1, 2, 3, 4, 7, 9, 10

Camino 2: 1, 2, 3, 5, 8, 10

Camino 3: 1, 2, 3, 4, 6, 8, 10

V (G) = Complejidad ciclomática.

A = Cantidad de aristas del grafo.

N = Cantidad de nodos del grafo.

R = Regiones en el grafo.

P = Nodos predicados. Nodos que tienen más de una arista de salida.

V (G) = A – N + 2

V (G) = 11 - 10 + 2

V (G) = 3

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

V (G) = Regiones (R)

V (G) = 3

La complejidad ciclomática del código es 3, lo que indica que hay 3 caminos independientes en el flujo del programa. Esto sugiere que el código tiene una complejidad moderada y puede ser probado adecuadamente con 3 casos de prueba para cubrir todos los caminos posibles.

Tabla 10: Caso de prueba Autenticar Usuario (Fuente: Elaboración propia).

Caso de prueba: Autenticar Usuario
Nombre de requisito: Autenticar.
Nombre de la persona que realiza la prueba: Alejandro Figueroa Rodríguez.
Descripción de la prueba: Prueba a la funcionalidad Autenticar usuario.
Entrada/Pasos de ejecución: Se introducen los siguientes datos para autenticar un usuario.
Datos:

- **username:** alejandrofr
- **password:** afr1231809

El usuario introduce estos datos y presiona el botón de acceder. Si los datos son correctos se autentica al usuario y lo deja entrar al sistema. En caso contrario de que los datos sean incorrectos se muestra un mensaje de error en la interfaz.

3.4.3 Prueba de Rendimiento.

Las pruebas de rendimiento según (Pressman, 2005) son esenciales para validar que el software cumpla con los requisitos bajo distintas condiciones de carga y uso. Su propósito es medir tiempos de respuesta, uso de recursos y capacidad de manejo de usuarios o transacciones. Pressman resalta su importancia para identificar cuellos de botella, optimizar el sistema y garantizar una experiencia confiable antes del despliegue.

A la aplicación se le realizó una prueba de rendimiento de tipo carga y estrés. Se utilizó la herramienta Apache JMeter (Ver Epígrafe 1.4.7), diseñando pruebas de cargas de comportamientos funcionales y la medición del rendimiento. El ambiente de prueba está conformado por:

Hardware de prueba:

1. Sistema Operativo: Windows 11 Pro 23h2.
2. Microprocesador: 4th Gen Intel(R) Core (TM) i3-4170 @ 3.70GHz.
3. Memoria RAM: 16 GB.
4. Disco Solido: 120 GB.
5. Tarjeta de video: Raedon RX 550 2Gigas.

Luego de haber definido el hardware se configuran los parámetros de Apache JMeter logrando un total de 500 usuarios conectados concurrentemente. En la tabla 12 se puede observar los resultados obtenidos por el sistema.

Parámetros que componen la tabla para una mejor interpretación:

- **Usuarios:** Total de usuarios de las pruebas.

- **Muestras:** El número de peticiones.
- **Media:** El promedio o la media de tiempo transcurrido en mili-segundos para las muestras de la URL dada.
- **Min:** El mínimo tiempo transcurrido en mili-segundos para las muestras de la URL dada.
- **Max:** El máximo tiempo transcurrido en mili-segundos para las muestras de la URL dada.
- **% Error:** Porcentaje de las peticiones con errores.
- **Rendimiento:** Rendimiento medido en base a peticiones por segundo/minuto/hora.
- **Kb/s Recibidos:** Rendimiento medido en Kbytes por segundos recibidos de las peticiones.

Tabla 11: Estadísticas resultantes de las pruebas de rendimiento utilizando el software JMeter (Fuente: Elaboración propia).

Usuarios	Muestra	Media	Min	Max	% Error	Rendimiento	KB/s Recibidos
200	200	233	12	450	0.00%	150.0/sec	699.34
500	500	1275	88	2151	0.00%	154.0/sec	718.87
500	2000	1536	68	3331	0.00%	220.8/sec	1029.06

La tabla muestra una media de tiempo de máximo 1.5 segundos aproximadamente para una carga de 500 usuarios haciendo 2000 peticiones en un solo segundo. Esto indica que el servidor web cumple con los requisitos no funcionales establecidos en el epígrafe 2.7.2.

Esta prueba de rendimiento ha sido realizada sobre el requisito funcional “Listar Configuraciones” y “Crear Configuraciones” para probar el rendimiento del servidor web demostrando que es completamente apto para un correcto funcionamiento en entornos de producción cumpliendo con los requisitos no funcionales de rendimiento establecidos.

Además de probar el rendimiento del servidor web también se llevaron a cabo pruebas de rendimiento al sistema de recomendación entrenado con los datos

establecidos en el epígrafe 3.3.1 y las configuraciones establecidas en el epígrafe 3.3.2. Se especifica de forma clara tanto los datos como las configuraciones utilizadas ya que el rendimiento de los modelos utilizados (Ver Epígrafe 2.1.2) depende directamente de ello además del hardware.

Para el proceso de actualización se utilizó un conjunto de 2000 datos extraídos del mismo dataset original de Picta. Estos datos son 1000 vistas (clicks) y 1000 likes que por supuesto no están dentro de los datos seleccionados para entrenamiento. Los resultados obtenidos del sistema fueron los siguientes:

- Tiempo aproximado de entrenamiento: 12 horas.
- Tiempo aproximado de actualización: 15 Minutos.
- Tiempo aproximado de inferencia o recomendación del sistema: 1 – 2.50 Segundos.

Con estos resultados se puede demostrar que este sistema de recomendación puede ser una solución a la problemática actual de rendimiento en los algoritmos de recomendación de las plataformas del proyecto Z17. Esto se debe a que utiliza técnicas como Fine Tunning (Ver Glosario de Términos página 101) o Tuneo Fino para actualizar el sistema sin tener que reentrenarlo completo cada vez que se creen nuevos datos. Además de que se puede mejorar el hardware utilizado para aumentar considerablemente la velocidad del tiempo de entrenamiento, algo que los algoritmos actuales no pueden hacer.

3.4.4 Pruebas de Seguridad.

La seguridad informática comprende la puesta en práctica de un conjunto de medidas preventivas y reactivas en los sistemas informáticos y tecnológicos, que posibilitan la protección de la información, persiguiendo como objetivo principal la integridad, confidencialidad y disponibilidad de la misma (Bautista, 2020). El software utilizado para la realización de las pruebas es Acunetix (Ver Epígrafe 1.4.7). Las pruebas realizadas se muestran a continuación:

- Ataques de inyección.
- Cross-Site Scripting (XSS).
- Falsificación de petición (CSRF).

- Autenticación y validación de roles.

Tabla 12: Resultados encontrados de las pruebas de seguridad realizadas (Fuente: Elaboración propia).

Tipo	Cantidad	Descripción	Recomendaciones
CSRF en API de recomendaciones.	1	El sistema no evalúa que solo las plataformas autorizadas puedan realizar las peticiones para generar recomendaciones.	Crear un sistema de autenticación y autorización de plataformas o sistemas externos para evitar intrusos no permitidos.
Ataques de inyección.	1	El sistema permite la inyección de código en la funcionalidad Recomendar.	Crear validaciones para los parámetros recibidos en las funciones y procesos que intervienen en el requisito Recomendar.

Resultado de las pruebas de seguridad:

Las pruebas de seguridad permitieron a través de un análisis del sistema en profundidad encontrar todas las vulnerabilidades que tenía el mismo pudiendo evitar futuros ataques y fallas al producto de software ya desplegado en producción. Estas pruebas son fundamentales ya que el sistema de recomendaciones puede manejar datos sensibles de cada plataforma y estas, además, utilizan al sistema de recomendaciones para mostrar un gran número de información en sus interfaces.

En relación con los requisitos no funcionales de seguridad descritos en el epígrafe 2.7.2, específicamente sobre la validación de roles en el sistema, se puede afirmar con certeza que este cumple plenamente con las especificaciones establecidas. El sistema distingue correctamente entre los dos roles definidos (Plataforma y Administrador), permitiendo a cada uno acceder únicamente a las funcionalidades descritas previamente en el epígrafe 2.3.

Los resultados obtenidos en las pruebas de seguridad fueron satisfactorios, ya que se logró solventar todos los problemas de seguridad encontrados llegando a la conclusión que el sistema de recomendación desarrollado es seguro y está en condiciones de ser utilizado por el cliente.

3.5 Validación de la investigación.

Para la validación de la hipótesis se aplicó además la técnica IADOV, la cual permitió evaluar el nivel de satisfacción de potenciales usuarios con el modelo propuesto (Mata Arias, 2017). Esta consistió en realizar una encuesta a una muestra de 15 usuarios (Ver Anexo 2) de entre los cuales 10 poseen experiencia de informática y 5 son usuarios con poca experiencia de informática.

Tabla 13: Cuadro lógico de IADOV (Mata Arias, 2017).

	3. ¿Considera relevantes las recomendaciones obtenidas?								
6. ¿Le satisfacen las recomendaciones dadas a partir de las salidas que provee?	SI			NO			NO SE		
	5. ¿Cree usted que este sistema será útil para las plataformas del proyecto Z17?								
	SI	NO	NO SE	SI	NO	NO SE	SI	NO	NO SE
Me satisface mucho	1	2	6	2	2	6	6	6	6
No me satisface tanto	2	2	3	2	3	3	6	3	6
Me da lo mismo 3	3	3	3	3	3	3	3	3	3
Me insatisface más de lo que me satisface	6	3	6	3	4	4	3	4	4
No me satisface nada	6	6	6	6	4	4	6	4	5
No lo sé	2	3	6	3	3	3	6	3	4

En la tabla anterior, las filas subrayadas corresponden a las preguntas cerradas de la encuesta. El número resultante de la interrelación de las tres preguntas indica la posición de cada sujeto en la escala de satisfacción, siendo esta la que se presenta a continuación:

1. Clara satisfacción.
2. Más satisfecho que insatisfecho.
3. No definida.
4. Más insatisfecho que satisfecho.
5. Clara insatisfacción.
6. Contradictoria.

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y - 1 de la siguiente forma:

Tabla 14: Índices de satisfacción (Mata Arias, 2017).

+1	Máximo de satisfacción
0.5	Más satisfecho que insatisfecho
0	No definido y contradictorio
-0.5	Mas insatisfecho que satisfecho
-1	Máxima insatisfacción

La satisfacción grupal se calcula por la siguiente fórmula (Mata Arias, 2017):

$$\mathrm{ISG} = \frac{\mathrm{A}(+1) + \mathrm{B}(0.5) + \mathrm{C}(0) + \mathrm{D}(-0.5) + \mathrm{E}(-1)}{\mathrm{N}}$$

En esta fórmula A, B, C, D, E, representan el número de sujetos con índice individual 1; 2; 3 o 6; 4; 5 y donde N representa el número total de sujetos del grupo. Para un total de 15 usuarios el resultado es el siguiente:

Tabla 15: Resultados de la técnica IADOV (Fuente: Elaboración propia).

Resultado	**Cantidad**	**%**
Total de usuarios de la muestra	15	100
Máximo de satisfacción	12	80

Más satisfecho que insatisfecho	2	13.33
No definida	0	0
Más insatisfecho que satisfecho	1	6.67
Clara insatisfacción	0	0
Contradictoria	-	-

Tabla 16: Variables de la formula ISG (Fuente: Elaboración propia).

A	12
B	2
C	0
D	1
E	0
N	15

$$\text{ISG} = \frac{12(+1) + 2(0.5) + 0(0) + 1(-0.5) + 0(-1)}{15}$$

$$ISG = 0.83$$

Conclusiones del método:

- Se obtuvieron criterios positivos expresados por unanimidad o mayoría.
- No existe contradicción en los resultados arrojados por el método aplicado.
- Dado un 83% de satisfacción general a la muestra de usuarios de prueba se constata la capacidad del sistema de recomendación de información basado en Deep Learning para mejorar la personalización y visibilidad en las plataformas del proyecto Z17.

3.6 Conclusiones del capítulo.

Como parte del desarrollo del presente capítulo se determinan las siguientes conclusiones parciales:

- ✓ El diseño del diagrama de despliegue facilitó la visión en cuanto a composición física y lógica del sistema.

- ✓ Aplicar los estándares de codificación permitió obtener en el sistema un código legible, estándar y fácil de comprender lo que asegura la calidad y facilita un futuro mantenimiento.

- ✓ El proceso de validación de la solución propuesta a través de las pruebas realizadas: caja negra, caja blanca, rendimiento y seguridad, permitieron identificar y corregir las no conformidades detectadas para obtener un producto de calidad.

- ✓ La aplicación de la técnica de IADOV, aportó los elementos sustanciales que permitieron validar la solución de la presente investigación y con ello la factibilidad del sistema implementado.

CONCLUSIONES GENERALES

1. El estudio de los referentes teóricos y el análisis de los diferentes sistemas de recomendaciones permitió determinar la no existencia de un sistema informático que responda a las necesidades requeridas por el cliente.

2. El diseño de la propuesta de solución permitió generar los artefactos más significativos de acuerdo con la metodología de desarrollo de software AUP-UCI Escenario 2 tomándose como referencia los requisitos detectados.

3. Se implementó un sistema de recomendación basado en Deep Learning que tiene en cuenta las preferencias de los usuarios y genera recomendaciones personalizadas para las plataformas del proyecto Z17.

4. Las técnicas de validación definidas para la propuesta de solución permitió la detección y corrección de las no conformidades detectadas y evidencian que el sistema constituye una solución funcional.

5. La validación del problema de investigación mediante la técnica IADOV demuestra que el sistema de recomendación desarrollado tiene en cuenta las preferencias de los usuarios y genera recomendaciones personalizadas para las plataformas del proyecto Z17.

RECOMENDACIONES

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

- ✓ Probar el sistema con el resto de plataformas del proyecto Z17 que por problemas de tiempo no se pudo en esta investigación. Las plataformas Apklis y toDus.
- ✓ Aumentar la cantidad de datos a utilizar para entrenar los modelos y mejorar el hardware utilizado en el proceso de entrenamiento para mejorar tanto el rendimiento como la precisión.

ANEXOS

Anexo 1: Preguntas concernientes a la entrevista realizada al cliente en el proceso de análisis e identificación de requisitos.

Estimado:

Usted ha sido seleccionado para colaborar, en su calidad de cliente, mediante esta entrevista en el proceso de análisis e identificación de requisitos para el desarrollo e implementación del sistema de recomendación basado en Deep Learning en las plataformas del proyecto Z17 (Apklis, toDus y Picta). Su colaboración será de gran ayuda para asegurar que el sistema responda a las necesidades y objetivos de la organización.

Por favor, sería de gran provecho que pudiera responder a las siguientes preguntas:

1. ¿Cuál es el principal objetivo que desean lograr con el sistema de recomendación en cada una de las plataformas del proyecto Z17?
2. ¿Qué funcionalidades principales debe ofrecer el SR para las plataformas?
3. ¿Qué tipo de contenido o elementos considera prioritario recomendar en cada plataforma?

    - Para Apklis: Todas las aplicaciones, aplicaciones más populares, personalizadas, o de una categoría específica.
    - Para toDus: Contactos, grupos o canales de interés.
    - Para Picta: Videos, transmisiones en vivo o categorías específicas.

4. ¿Cómo deben de interactuar los usuarios finales con el sistema a través de las plataformas y que resultados esperan obtener de las recomendaciones?
5. ¿Qué nivel de personalización esperan para las recomendaciones (individuales, grupales o generales para todos los usuarios)?
6. ¿Qué tipo de respuesta debe generar el SR para las plataformas (documentos completos, identificadores únicos o resúmenes)?

7. ¿Qué funcionalidades específicas deberá incluir la interfaz administrativa para gestionar el sistema?
8. ¿Qué tipo de datos sobre los usuarios están disponibles actualmente en cada plataforma? (historial de navegación, preferencias declaradas, acciones previas).
9. ¿Qué nivel de rendimiento espera que tenga el sistema de recomendación en términos de usuarios simultáneos que puedan interactuar con las plataformas del proyecto Z17 de manera eficiente?

Anexo 2: Validación mediante la técnica IADOV (Encuesta).

Estimado usuario:

La siguiente encuesta de carácter anónimo nos permitirá familiarizarnos con su conocimiento y postura con respecto al sistema de recomendación basado en Deep Learning para las plataformas del proyecto Z17. Le agradecemos que lea cuidadosamente cada pregunta antes de responder y marque con una X en las opciones que considere más acertadas. Su opinión será de gran importancia, pues permitirá evaluar la correcta implementación que supone el sistema de recomendación en cuestión.

Edad_____ Sexo_____ Nivel escolar_____

1. ¿Está familiarizado usted con la recomendación de información en plataformas digitales?
   SÍ_____ NO_____ NO SÉ_____
2. ¿Considera usted que las recomendaciones tributan a obtener resultados adecuados a sus intereses?
   SÍ_____ NO_____ NO SÉ_____
3. ¿Considera relevantes las recomendaciones obtenidas?
   SÍ_____ NO_____ NO SÉ_____
4. ¿Cree importante contar con la categorización de las recomendaciones obtenidas?
   MUY NECESARIO_____ NECESARIO_____ POCO NECESARIO_____ INECESARIO_____

5. ¿Cree usted que este Sistema de recomendación será útil para las plataformas del proyecto Z17?
   SÍ_____ NO_____ NO SÉ_____
6. ¿Le satisfacen las recomendaciones dadas a partir de las salidas que provee?
   Me satisface mucho_____ No me satisface tanto_____ Me da lo mismo_____ Me insatisface más de lo que me satisface_____ No me satisface nada_____ No lo sé_____

Anexo 3: Guía de Observación.

1. Observar y analizar la interacción de los usuarios con el sistema de recomendación en las plataformas del proyecto Z17 para evaluar la personalización y relevancia de las recomendaciones, así como la satisfacción del usuario.
2. La observación se llevará a cabo durante sesiones de prueba en un entorno controlado, donde los usuarios interactúan con las plataformas para las que fue diseñado el sistema de recomendación.
3. Los observadores serán miembros del equipo de investigación, que actuarán de manera neutral, registrando la interacción de los usuarios sin intervenir directamente.
4. Variables a Observar: personalización, relevancia de las recomendaciones y satisfacción del usuario.

Los procesos estudiados incluyen: Algoritmos de recomendación, personalización y adaptación del sistema a las preferencias del usuario.

Los fenómenos observados abarcan: Personalización de las recomendaciones, satisfacción del usuario y precisión del sistema.

Las características fenomenológicas del objeto de estudio incluyen la usabilidad del sistema, la relevancia y personalización de las recomendaciones. Estas observaciones permitieron describir y explicar cómo estos elementos influyen en la calidad del sistema y someterlos a una elaboración racional para mejorar su diseño y funcionalidad.

GLOSARIO DE TÉRMINOS

Deep Learning: Es un sub campo del aprendizaje automático (Machine Learning) que utiliza redes neuronales profundas para modelar y resolver problemas complejos. Estas redes se componen de múltiples capas que procesan grandes volúmenes de datos, lo que permite que el modelo aprenda representaciones y características jerárquicas de la información de manera autónoma (Sharifani & Amini, 2023).

Personalización: Es el proceso de adaptar los servicios, contenidos o productos ofrecidos a un usuario, basándose en sus preferencias, comportamientos previos o datos históricos. En el contexto de sistemas de recomendación, la personalización implica proporcionar recomendaciones ajustadas a los intereses de cada usuario individual.

Preferencias de los usuarios: Son los intereses o elecciones individuales de los usuarios basadas en sus interacciones previas con el sistema. Estas preferencias se pueden derivar de la actividad de navegación, calificaciones, compras anteriores, entre otros factores, y son fundamentales para generar recomendaciones personalizadas (Roy & Dutta, 2022).

Benchmarking: Es el proceso de comparar el desempeño de un sistema, algoritmo o modelo con otros existentes, generalmente en un conjunto estándar de datos o métricas. En el contexto de sistemas de recomendación, el benchmarking se utiliza para evaluar y mejorar la precisión y eficiencia del sistema (Francis & Holloway, 2007).

Hiperparámetros: Son los parámetros que controlan el proceso de entrenamiento de un modelo de aprendizaje automático. A diferencia de los parámetros internos (como los pesos en una red neuronal), los hiperparámetros deben ser definidos antes del entrenamiento y pueden incluir la tasa de aprendizaje, el número de capas de la red, entre otros (Kammoun et al., 2022).

Sesgos: En el contexto de sistemas de recomendación y aprendizaje automático, los sesgos son desviaciones sistemáticas que afectan las decisiones

del modelo. Pueden surgir debido a datos desbalanceados, decisiones previas del usuario o limitaciones en los algoritmos (J. Chen et al., 2023).

Candidatos: Son las opciones o ítems que un sistema de recomendación considera para ser recomendados a un usuario (Covington et al., 2016).

API (Application Programming Interface): Es un conjunto de reglas y protocolos que permiten que diferentes aplicaciones o sistemas se comuniquen entre sí (Ofoeda et al., 2019).

Fine Tunning: En él aprendizaje profundo, el ajuste fino o Fine Tunning es un enfoque para transferir el aprendizaje en el que los pesos de un modelo previamente entrenado se entrenan con datos nuevos. En el contexto de los sistemas de recomendaciones a medida que se recopilan nuevos datos de los usuarios el modelo puede ser así ajustado y refinado para generar nuevas recomendaciones en base a las nuevas preferencias de los usuarios (Käding et al., 2017).

REFERENCIAS BILBIOGRÁFICAS

Acosta, G. (2020, mayo 21). Modelos de recomendación: Recomendando qué recomendar. *Quanam*. https://quanam.com/modelos-de-recomendacion-recomendando-que-recomendar/.

Acunetix | Web Application and API Security Scanner. (2024). Acunetix. https://www.acunetix.com/.

Alamdari, P. M., Navimipour, N. J., Hosseinzadeh, M., Safaei, A. A., & Darwesh, A. (2020). A Systematic Study on the Recommender Systems in the E-Commerce. *IEEE Access*, *8*, 115694-115716. IEEE Access. https://doi.org/10.1109/ACCESS.2020.3002803.

Alfaro, M., & Roberto, C. (2022). *«EL LENGUAJE PYTHON Y SU POTENCIAL EN EL DESARROLLO DE SOFTWARE DE INTELIGENCIA ARTIFICIAL».* | *EBSCOhost*. https://openurl.ebsco.com/EPDB%3Agcd%3A6%3A14702230/detailv2?sid=ebsco%3Aplink%3Ascholar&id=ebsco%3Agcd%3A156756997&crl=c.

Almaraz Pérez, A. (2013, septiembre 25). *Sistemas de recomendación*. Universidad Autónoma Metropolitana. https://doi.org/10.24275/uami.q811kj77c.

Apache JMeter—Apache JMeter™. (2024). https://jmeter.apache.org/.

Bautista, E. C. R. (2020). *GUÍA DE PRINCIPIOS Y BUENAS PRÁCTICAS PARA PRUEBAS DE SEGURIDAD DE SOFTWARE EN APLICACIONES WEB PARA UNA EMPRESA DEL SECTOR PRIVADO*.

Betru, B. T., Onana, C. A., & Batchakui, B. (2017). Deep Learning Methods on Recommender System: A Survey of State-of-the-art. *International Journal of Computer Applications*, *162*(10).

Botero Tabares, R. (2023). *Grasp Patterns and Anti-Patterns: An Object Oriented Approach from Logic Programming*. http://hdl.handle.net/10785/13571.

Casalegno, F. (2022, noviembre 25). *Recommender Systems—A Complete Guide to Machine Learning Models | by Francesco Casalegno | Towards Data Science*. Recommender Systems — A Complete Guide to Machine Learning Models. https://towardsdatascience.com/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748.

Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., & He, X. (2023). Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Trans. Inf. Syst.*, *41*(3), 67:1-67:39. https://doi.org/10.1145/3564284.

Covington, P., Adams, J., & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 191-198. https://doi.org/10.1145/2959100.2959190.

FastAPI. (2024). https://fastapi.tiangolo.com/.

Francis, G., & Holloway, J. (2007). What have we learned? Themes from the literature on best-practice benchmarking. *International Journal of Management Reviews*, *9*(3), 171-189. https://doi.org/10.1111/j.1468-2370.2007.00204.x.

Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., & Li, Y. (2023). A Survey of Graph Neural Networks for Recommender

Systems: Challenges, Methods, and Directions. *ACM Trans. Recomm. Syst.*, *1*(1), 3:1-3:51. https://doi.org/10.1145/3568022.

Ghosh, K., Bellinger, C., Corizzo, R., Branco, P., Krawczyk, B., & Japkowicz, N. (2024). The class imbalance problem in deep learning. *Machine Learning*, *113*(7), 4845-4901. https://doi.org/10.1007/s10994-022-06268-8.

Gong, M., & Zhernov, A. (2024, septiembre 5). *Advanced machine learning helps Play Store users discover personalised apps*. Google DeepMind. https://deepmind.google/discover/blog/advanced-machine-learning-helps-play-store-users-discover-personalised-apps/.

González Matos, M. D. (2021). *Sistema de gestión y aseguramiento material para la Organización Nacional de Bufetes Colectivos* [bachelorThesis, Universidad de las Ciencias Informáticas. Facultad 1]. https://repositorio.uci.cu/jspui/handle/123456789/10406.

Guerrero, C. A., Suárez, J. M., & Gutiérrez, L. E. (2013). Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. *Información tecnológica*, *24*(3), 103-114. https://doi.org/10.4067/S0718-07642013000300012.

Gutierrez, J. C. J. (2023, junio 29). Los Sistemas de Recomendación y la Ciencia de Datos. *Medium*. https://medium.com/@jcjerez_77135/los-sistemas-de-recomendaci%C3%B3n-y-la-ciencia-de-datos-1b2fa965f47b.

Iglesias Mizrahi, A. (2016). *Desarrollo del Sistema de Recomendación de equipos de investigación para tesis de grado* [bachelorThesis, Universidad de las

Ciencias Informáticas. Facultad 3]. https://repositorio.uci.cu/jspui/handle/123456789/7728.

Käding, C., Rodner, E., Freytag, A., & Denzler, J. (2017). Fine-Tuning Deep Neural Networks in Continuous Learning Scenarios. En C.-S. Chen, J. Lu, & K.-K. Ma (Eds.), *Computer Vision – ACCV 2016 Workshops* (pp. 588-605). Springer International Publishing. https://doi.org/10.1007/978-3-319-54526-4_43.

Kammoun, A., Slama, R., Tabia, H., Ouni, T., & Abid, M. (2022). Generative Adversarial Networks for face generation: A survey. *ACM Computing Surveys*, 1122445.1122456. https://doi.org/10.1145/1122445.1122456.

Keras. (2024). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Keras&oldid=160878252.

Kyurkchiev, N., & Markov, S. (2015). *SIGMOID FUNCTIONS: SOME APPROXIMATION, AND MODELLING ASPECTS*.

Mata Arias, L. A. (2017). *Subsistema de recomendación de información para el buscador cubano Orión* [bachelorThesis, Universidad de las Ciencias Informáticas. Facultad 1]. https://repositorio.uci.cu/jspui/handle/123456789/8041

Microsoft. (2024). *Visual Studio Code—Code Editing. Redefined*. https://code.visualstudio.com/.

MongoDB: La plataforma de datos para desarrolladores. (2024). MongoDB. https://www.mongodb.com/.

NumPy—About Us. (2024). https://numpy.org/about/.

Ofoeda, J., Boateng, R., & Effah, J. (2019). Application Programming Interface (API) Research: A Review of the Past to Inform the Future. *International Journal of Enterprise Information Systems (IJEIS)*, *15*(3), 76-95. https://doi.org/10.4018/IJEIS.2019070105.

Pandas. (2024). https://pandas.pydata.org/about/index.html.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan..

Python. (2024). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Python&oldid=161998593.

Roy, D., & Dutta, M. (2022). A systematic review and research perspective on recommender systems. *Journal of Big Data*, *9*(1), 59. https://doi.org/10.1186/s40537-022-00592-5.

Ruiz Ricardo, B., & Vaillant Valdéz, C. de la C. (2015). *Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0* [bachelorThesis, Universidad de las Ciencias Informáticas. Facultad 2]. https://repositorio.uci.cu/jspui/handle/123456789/7178.

Sharifani, K., & Amini, M. (2023). *Machine Learning and Deep Learning: A Review of Methods and Applications* (SSRN Scholarly Paper No. 4458723). https://papers.ssrn.com/abstract=4458723.

SOMMERVILLE, I. (2005). *Ingenieria del Software*.

Steck, H., Baltrunas, L., Elahi, E., Liang, D., Raimond, Y., & Basilico, J. (2021). Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine*, *42*(3), Article 3. https://doi.org/10.1609/aimag.v42i3.18140

TensorFlow. (2024). https://www.tensorflow.org/

TensorFlow Recommenders. (2024). TensorFlow. https://www.tensorflow.org/recommenders?hl=es-419

Tomás Cruz, A. (2024). *Reconocimiento de lenguaje mediante un clasificador multiclase SoftMax*. https://hdl.handle.net/20.500.12371/21553

Tsui, F., Karam, O., & Bernal, B. (2022). *Essentials of Software Engineering*. Jones & Bartlett Learning.

Varela, D., Aguilar, J., Montoya, E., & Monsalve-Pulido, J. (2020). *Propuesta Arquitectónica de un Sistema de Recomendación Híbrido Adaptativo*.

Xia, J. H. H. (2023). *DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA UNA EMPRESA DE SERVICIOS ONLINE*.

Xie, R., Ling, C., Wang, Y., Wang, R., Xia, F., & Lin, L. (2020). Deep Feedback Network for Recommendation. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2519-2525. https://doi.org/10.24963/ijcai.2020/349

Z17. (2024). https://z17.cu/

Zhang, Q., Lu, J., & Zhang, G. (2021). *Recommender Systems in E-learning*. https://www.oaepublish.com/articles/jsegc.2020.06