

**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(es):**

Alejandro Figueroa Rodríguez

**Tutor(es):**

Yadier Perdomo Cuevas

Aneyty Martin García

**Co-tutor:** <nombre co-tutor (opcional de existir)>

**Consultante:** <nombre consultante (opcional de existir)>

**Asesor:** <nombre asesor (opcional de existir)>

La Habana, <mes> de 2024

Año 66 de la Revolución

**DECLARACIÓN DE AUTORÍA**

El autor Alejandro Figueroa Rodríguez del trabajo de diploma con título ***“Desarrollo e Implementación de un Sistema de Recomendación Basado en Deep Learning para el 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** | **<nombre del autor>** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Autor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Autor |
| **<nombre del tutor>** | **<nombre del tutor>** |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Firma del Tutor |

**Agradecimientos**

**Dedicatoria**

**Resumen**

El presente trabajo tiene como objetivo desarrollar un sistema de recomendación que mejore la personalización y visibilidad del contenido presente en 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 4 generándose 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 para el usuario.

Palabras claves

Deep Learning, Aprendizaje Profundo, Inteligencia Artificial, Redes Neuronales, proyecto Z17, Sistema de recomendación,

***Abstract***

**Tabla de contenidos**

[Introducción 12](#_Toc180027525)

[Capítulo 1: Fundamentos y referentes teórico-metodológicos sobre el objeto de estudio. 18](#_Toc180027526)

[1.1 Sistema de recomendación. 18](#_Toc180027527)

[1.1.1 Clasificación de los Sistemas de Recomendaciones. 19](#_Toc180027528)

[Basados en Popularidad. 20](#_Toc180027529)

[Basados en Contenido. 21](#_Toc180027530)

[Basados en Filtrado Colaborativo. 21](#_Toc180027531)

[Híbridos. 23](#_Toc180027532)

[Filtrado Demográfico. 23](#_Toc180027533)

[Recomendadores Conversacionales. 24](#_Toc180027534)

[1.1.2 Retroalimentación 25](#_Toc180027535)

[1.2 Estudio de Sistemas Homólogos. 27](#_Toc180027536)

[1.2.1 Estudio de Sistemas Homólogos en el ámbito Internacional. 28](#_Toc180027537)

[1.2.2 Estudio de Sistemas Homólogos en el ámbito Nacional. 32](#_Toc180027538)

[1.2.3 Conclusión de los sistemas homólogos. 34](#_Toc180027539)

[1.3 Propuesta de solución. 35](#_Toc180027540)

[1.3.1 Recomendadores basados en Redes Neuronales Profundas. 35](#_Toc180027541)

[1.3.2 Ventajas de utilizar un sistema de recomendación basado en Redes Neuronales Profundas. 36](#_Toc180027542)

[1.3.3 Desafíos 37](#_Toc180027543)

[1.4 Tecnologías y herramientas para el desarrollo. 39](#_Toc180027544)

[1.4.1 Metodología de desarrollo de software. 39](#_Toc180027545)

[1.4.2 Lenguaje de Programación. 42](#_Toc180027546)

[Python V 3.11.9 42](#_Toc180027547)

[Ventajas de Python 43](#_Toc180027548)

[1.4.3 Sistema gestor de base de datos. 43](#_Toc180027549)

[PostgreSQL V 15.2 43](#_Toc180027550)

[1.4.4 Bibliotecas. 43](#_Toc180027551)

[Tensorflow V 2.15.0 44](#_Toc180027552)

[Keras V 2.15.0 45](#_Toc180027553)

[TensorFlow Recommenders V 0.7.3 45](#_Toc180027554)

[NumPy V 1.26.4 46](#_Toc180027555)

[Pandas V 2.2.3 46](#_Toc180027556)

[1.4.5 Entorno de Desarrollo. 47](#_Toc180027557)

[Visual Studio Code V 1.92.2 47](#_Toc180027558)

[1.4.6 Servidor Web. 47](#_Toc180027559)

[Fastapi V 0.115.0 47](#_Toc180027560)

[1.5 Conclusiones del Capítulo. 48](#_Toc180027561)

[Capítulo 2: Análisis y diseño del sistema para las recomendaciones en las plataformas del proyecto z17. 50](#_Toc180027562)

[2.1 Descripción de la Propuesta de Solución. 50](#_Toc180027563)

[2.1.1 Etapas. 50](#_Toc180027564)

[2.1.2 Modelos. 52](#_Toc180027565)

[2.2 Modelo Conceptual 57](#_Toc180027566)

[2.3 Diagrama de caso de uso del sistema. 58](#_Toc180027567)

[2.4 Diagrama de clases de diseño. 58](#_Toc180027568)

[2.5 Diagrama de secuencia. 59](#_Toc180027569)

[2.6 Modelo de datos 60](#_Toc180027570)

[2.7 Requisitos de la propuesta de solución 60](#_Toc180027571)

[2.7.1 Requisitos Funcionales 61](#_Toc180027572)

[2.7.2 Requisitos no funcionales 61](#_Toc180027573)

[2.8 Arquitectura de software 63](#_Toc180027574)

[2.9 Patrones de Diseño 65](#_Toc180027575)

[2.9.1 Patrones GRASP 65](#_Toc180027576)

[Experto 65](#_Toc180027577)

[Creador 66](#_Toc180027578)

[Alta cohesión 67](#_Toc180027579)

[Bajo acoplamiento 68](#_Toc180027580)

[2.9.2 Patrones GOF 70](#_Toc180027581)

[Singleton 70](#_Toc180027582)

[Template Method 71](#_Toc180027583)

[2.10 Conclusiones del capítulo 72](#_Toc180027584)

[Referencias Bilbiográficas 73](#_Toc180027585)

**Índice de tablas**

[Tabla 1 Fases de las variaciones de AUP para la UCI (González Matos, 2021). 39](#_Toc180027590)

[Tabla 5 Descripción Requisitos Funcionales 60](#_Toc180027591)

[Tabla 6 Descripción de los Requisitos no Funcionales 60](#_Toc180027592)

**Índice de figuras**

[Figura 1 Clasificación de sistemas de recomendaciones (Gutierrez, 2023). 19](#_Toc180027593)

[Figura 2 Filtrado Basado en Contenido (Xia, 2023). 20](#_Toc180027594)

[Figura 3 Filtrado Colaborativo basado en usuarios (Acosta, 2020). 21](#_Toc180027595)

[Figura 4 Filtrado Colaborativo basado en elementos (Xia, 2023). 22](#_Toc180027596)

[Figura 5 Clasificación de los Recomendadores Conversacionales (Xia, 2023). 24](#_Toc180027597)

[Figura 6 Retroalimentación Explicita vs Implícita en sistemas de recomendaciones (Casalegno, 2022) 26](#_Toc180027598)

[Figura 7 Arquitectura del Sistema de Recomendación de Youtube (Covington et al., 2016) 28](#_Toc180027599)

[Figura 8 Arquitectura de una red neuronal profunda llamada Modelo de dos torres para recomendación (Kammoun et al., 2022). 35](#_Toc180027600)

[Figura 9 Metodología AUP versión UCI escenario 1. 40](#_Toc180027601)

[Figura 10 Metodología AUP versión UCI escenario 2. 40](#_Toc180027602)

[Figura 11 Metodología AUP versión UCI escenario 3. 40](#_Toc180027603)

[Figura 12 Metodología AUP versión UCI escenario 4. 40](#_Toc180027604)

[Figura 13 Arquitectura de capas del sistema. 51](#_Toc180027605)

[Figura 14 Estructura típica de un sistema de recomendación (Gao et al., 2023). 51](#_Toc180027606)

[Figura 15 Arquitectura de red neuronal de dos torres. 52](#_Toc180027607)

[Figura 16 Las arquitecturas NDR (neural deep retrieval), como los codificadores de dos torres, son conceptualmente similares a los modelos de factorización. 53](#_Toc180027608)

[Figura 17 Función de activación softmax (Tomás Cruz, 2024). 54](#_Toc180027609)

[Figura 18 Arquitectura del modelo de recuperación (Covington et al., 2016) 55](#_Toc180027610)

[Figura 19 Arquitectura de un modelo de clasificación (Covington et al., 2016) 56](#_Toc180027611)

[Figura 20 Modelo Conceptual 56](#_Toc180027612)

[Figura 21 Diagrama de caso de uso del sistema. 57](#_Toc180027613)

[Figura 22 Diagrama de clases con estereotipos web. 58](#_Toc180027614)

[Figura 23 Diagrama de secuencia. 58](#_Toc180027615)

[Figura 24 Modelo de datos. 59](#_Toc180027616)

[Figura 25 Clase del modelo de recuperación donde se refleja el patrón Experto. 65](#_Toc180027617)

[Figura 26 Acción Entrenar encargada de crear instancias de objetos 66](#_Toc180027618)

[Figura 27 Clase DataPipeline utilizando el patrón alta cohesión. 68](#_Toc180027619)

[Figura 28 Clase ModelConfig utilizando el patrón bajo acoplamiento. 69](#_Toc180027620)

[Figura 29 Creación de dos instancias globales en el sistema. 70](#_Toc180027621)

[Figura 30 Clase interfaz que se utiliza como platilla para la clase DataPipeline. 70](#_Toc180027622)

# 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*, s. f.).

Entre sus principales soluciones se encuentran (*Z17*, s. f.):

* 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 y precisas, 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.

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 su 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 comúnmente como sistema de recomendación basado en contenido.

Este proceso que realiza Picta para generar nuevas recomendaciones puede tardar hasta tres días por cada nueva generación de candidatos. Esto es bastante lento y poco sostenible en el tiempo ya que al aumentar los datos de la plataforma esta generación de recomendaciones también aumenta considerablemente.

Actualmente se utilizan algoritmos convencionales basados en estadística lo cual es bastante útil para volúmenes de datos pequeños o medianos pero se quedan cortos para este tipo de plataformas con un contenido considerable y en rápido 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 contribuir al mejoramiento de la personalización y visibilidad del contenido en las plataformas del proyecto Z17?

Se plantea como **objetivo general**: Desarrollar un sistema de recomendaciones que contribuya al mejoramiento de la personalización y visibilidad del contenido en las plataformas del proyecto Z17.

Se establece como **campo de acción**: El Sistema de recomendación para las plataformas Picta, Todus y Apklis.

**Pregunta Científica:** ¿Un sistema de recomendación (SR) con redes neuronales profundas, podría mejorar la personalización y visibilidad del contenido presente en las plataformas del proyecto z17?

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

1. ¿Qué modelos existen actualmente para dar solución a la problemática dada?
2. ¿Qué modelo ha sido seleccionado para dar solución al problema?
3. ¿Cómo aplicar el modelo del sistema de recomendación propuesto?
4. ¿El modelo de recomendaciones seleccionado para las plataformas cumple con su propósito?

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

* Elaboración del estado del arte de los sistemas de recomendación y los principales conceptos y elementos teóricos del tema.
* Descripción y modelación de los artefactos generados durante el desarrollo de la solución propuesta.

* Implementación de las principales funcionalidades de la solución informática propuesta.
* Integración del sistema de recomendación con las plataformas del proyecto Z17.

* Análisis de los resultados arrojados de las pruebas realizadas para evaluar el rendimiento y la precisión del sistema de recomendación.

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 clave 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 redes neuronales profundas. Este enfoque permitió comprender cómo los principios y técnicas históricas han influido en el diseño y la implementación del sistema moderno.

**Hipotético-deductivo:** Permite reflejar los elementos comunes entre los fenómenos estudiados como:

* Tipos de algoritmos de recomendación.
* Comportamiento del usuario.
* Desempeño de los sistemas de recomendación.

Establecer generalizaciones y analizar cada uno de los detalles hasta establecer las relaciones existentes entre estos fenómenos.

**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). Permite, además, obtener información sobre la evolución y el estado actual del objeto que se investiga, tanto nacional como internacional. Sirve de referencia en la selección de materiales de estudio para la construcción de un sistema de recomendación.

**Metidos Empíricos**

**Técnica Iadov:** Este método se utiliza en la investigación para validar la retroalimentación de los usuarios respecto al nivel de satisfacción y la fiabilidad del SR en la selección de materiales de estudio. A través de esta técnica, se recopilan y analizan las opiniones de los usuarios para evaluar la efectividad 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.

Los procesos estudiados incluyen:

* Algoritmos de recomendación.
* Interacción del usuario con el sistema.
* Adaptación del sistema a las preferencias del usuario.

Los fenómenos observados abarcan

* Precisión de las recomendaciones.
* Satisfacción del usuario.
* Eficiencia 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 efectividad del sistema y someterlos a una elaboración racional para mejorar su diseño y funcionalidad.

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

**Capítulo 1:** Fundamentación teórica. 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:** Descripción de la propuesta de solución. En este capítulo se determinan los servicios que brindará el sistema, definiéndose las funcionalidades que debe cumplir, así como el diseño del mismo, y se generan los artefactos que propone la metodología.

**Capítulo 3:** Validación de la solución propuesta. 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, analizando las ventajas y limitaciones de cada enfoque así como una vista al uso de esto tanto en el ámbito nacional como internacional. A continuación, se enfocará en los algoritmos de última generación, como las redes neuronales profundas y los modelos híbridos, que han revolucionado el campo y han permitido alcanzar niveles de personalización sin precedentes (Betru et al., 2017). 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.

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 web conocidas que necesitan sistemas de recomendación eficientes para mantener el interés de los usuarios debido a la vasta diversidad de contenidos disponibles.

1. [**Youtube**](https://www.youtube.com/)**:** Cada minuto los usuarios suben [**500 horas de vídeos**](https://www.oberlo.com/blog/youtube-statistics) , es decir, un usuario tardaría 82 años en ver todos los vídeos subidos solo en la última hora.
2. [**Spotify**](https://www.spotify.com/)**:** Los usuarios pueden escuchar más de[**80 millones de canciones y podcasts**](https://newsroom.spotify.com/company-info/) .
3. [**Amazon**](https://www.amazon.com/)**:** Los usuarios pueden comprar más de[**350 millones de productos diferentes**](https://www.retailtouchpoints.com/resources/how-many-products-does-amazon-carry) .

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

### 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. En las siguientes secciones, se emprenderá un recorrido por algunos de estos algoritmos, explorando sus mecanismos y la forma en que han revolucionado la manera en que se interactúa con los datos y las decisiones que tomamos a partir de ellos.

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 recomendaciones (Gutierrez, 2023).

Los sistemas de recomendaciones se pueden clasificar en varias categorías. Estas categorías permiten entender mejor las características y aplicaciones de cada tipo de sistema.

#### 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).

#### Basados en Contenido.

Genera recomendaciones basadas en el conocimiento que se tiene sobre los elementos que el usuario ha valorado (implícita o explícitamente), recomendando elementos similares. 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).

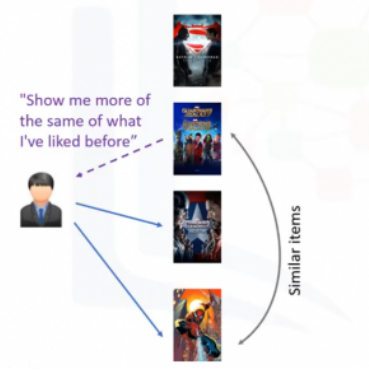


Figura 2 Filtrado Basado en Contenido (Xia, 2023).

#### Basados en Filtrado Colaborativo.

El filtrado colaborativo también se trata de uno de los métodos más comunes en los recomendadores. Dentro de este modelo se pueden encontrar las técnicas llamadas Memory-based, que utilizan toda la matriz de datos con sus calificaciones para generar una predicción. 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. Básicamente, el modelo buscará usuarios con gustos similares al del usuario objetivo (Véase la Figura 3), y recomendará productos que les hayan gustado a estos usuarios, ya que, si dos usuarios tienen gustos parecidos, seguramente les gusten los mismos productos. Para encontrar un conjunto de usuarios parecidos, se usan técnicas como: Busqueda de vecinos más cercanos o Nearest Neighbor Search (NNS).

La ventaja de este sistema es que es de fácil implementación y brindan un alto nivel de cobertura. Y además es capaz de capturar características sutiles y no requiere tener una compresión del contenido del ítem.

La desventaja es que tendrá dificultades en recomendar productos nuevos, ya que, al ser nuevo el ítem, este tendrá una falta de interacción con los usuarios. Esto también se aplica para los 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).

#### Híbridos.

Estos sistemas utilizan una combinación de los sistemas basados en filtrado colaborativo y filtrado basado en contenido. Se trata 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 híbrida pueden dividirse en dos grupos:

* **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 (Varela et al., 2020).

#### 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).

#### Recomendadores Conversacionales.

En este nuevo enfoque, según, 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).

Los recomendadores conversacionales son sistemas que guían al usuario a través del espacio de productos, ofreciendo sugerencias y solicitando feedback.

Algunos ejemplos destacados de recomendadores conversacionales son:

* Value Elicitation: Esta metodología se basa en un valor específico introducido por el usuario, lo que permite una búsqueda efectiva en función de una característica, como por ejemplo, un género = acción. La desventaja es que el usuario debe tener un amplio conocimiento sobre las características del producto o ítem a buscar, de lo contrario, este sistema resulta ineficaz.
* Ratings-based feedback: A diferencia del enfoque anterior, esta metodología no requiere un gran conocimiento por parte del usuario sobre las características del producto que busca. Los usuarios asignan una calificación sencilla, como por ejemplo, 4 estrellas de 5, para indicar su satisfacción con la recomendación. En este enfoque, el usuario no devuelve una retroalimentación detallada sobre las características del producto.
* Preference-based feedback: Esta metodología el usuario indica su recomendación preferida, en vez de clasificar un determinado conjunto de recomendaciones. Se trata de un enfoque de muy bajo coste de retroalimentación por parte del usuario, y además requiere de un mínimo conocimiento de dominio, solamente la capacidad de distinguir si una recomendación es mala o buena.
* Critiquing: Los usuarios dan retroalimentaciones específicas sobre las recomendaciones. Donde expresan sus preferencias, en lugar de simplemente aceptar o rechazar. Por ejemplo, retroalimentaciones de este estilo “Dame más productos de este estilo, pero …” (Xia, 2023).



Figura 5 Clasificación de los Recomendadores Conversacionales (Xia, 2023).

Tras analizar los algoritmos previos, se ha optado por implementar un sistema de recomendación híbrido que integra las fortalezas del filtrado basado en contenido y el colaborativo. Este enfoque híbrido supera las limitaciones de los sistemas actuales en las plataformas como Picta y Apklis al aprovechar la retroalimentación directa de los usuarios, lo cual permite refinar las recomendaciones de manera continua y dinámica.

### 1.1.2 Retroalimentación

Como algunas personas creen, “la red nos escucha”, y es que a través de todas las interacciones que hacemos en las plataformas proveemos información de forma directa (**explícito**), por ejemplo, cuando otorgamos una manito arriba a un producto, o de forma indirecta (**implícita**), cuando por ejemplo, simplemente vemos un video. Toda esta información es almacenada para analizar nuestras preferencias y luego con base en diferentes algoritmos poder sugerirnos productos o servicios que potencialmente pueden ser de nuestro interés (Xie et al., 2020).

Algunos ejemplos de recolección de datos de forma **explícitas** son:

* Solicitar al usuario que pondere sobre la base de una escala proporcionada, algún tema en particular.
* Solicitar al usuario que pondere un conjunto de temas de una lista de temas favoritos.
* 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:

* 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.
* Obtener un listado de los artículos que el usuario ha seleccionado o visto en su computadora.
* Analizar las redes sociales de las que el usuario forma parte y de esta manera conocer sus gustos y preferencias (Roy & Dutta, 2022).



Figura 6 Retroalimentación Explicita vs Implícita en sistemas de recomendaciones (Casalegno, 2022)

## 1.2 Estudio de Sistemas Homólogos.

En la presente investigación, se han analizado diversos sistemas recomendadores 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:

* Novedad Tecnológica: Evalúa qué tan reciente y avanzada es la tecnología utilizada.
* Tipo de Algoritmo: Determina qué tipo de algoritmo fue utilizado para las recomendaciones.
* Escalabilidad: La capacidad del sistema para manejar un gran número de usuarios y ítems sin degradar el rendimiento.
* Personalización: Este criterio mide qué tan bien las recomendaciones se ajustan a las preferencias y comportamientos individuales de los usuarios.

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

### 1.2.1 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).

Recomendar vídeos de YouTube es un gran desafío desde tres grandes perspectivas:

* **Escala:** Muchos algoritmos de recomendación funcionan bien en problemas pequeños, pero no a gran escala. Se necesitan algoritmos de aprendizaje distribuido y sistemas de servicio eficientes para manejar la gran base de usuarios y el vasto corpus de YouTube.
* **Frescura:** YouTube tiene un corpus dinámico con muchas horas de vídeo subidas por segundo. El sistema de recomendación debe ser responsivo para modelar contenido nuevo y las últimas acciones del usuario, equilibrando entre contenido nuevo y videos establecidos.
* **Ruido:** El comportamiento histórico de los usuarios en YouTube es difícil de predecir debido a la escasez de datos y factores externos no observables. Los algoritmos deben ser robustos para manejar señales de retroalimentación implícitas o ruidosas y metadatos mal estructurados.

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. El sistema está construido en Google Brain que recientemente fue de código abierto como TensorFlow. TensorFlow proporciona un marco flexible para experimentar con varias arquitecturas de redes neuronales profundas que utilizan entrenamiento distribuido a gran escala. Sus modelos aprenden aproximadamente mil millones de parámetros y están entrenados en cientos de miles de millones de ejemplos (Covington et al., 2016).

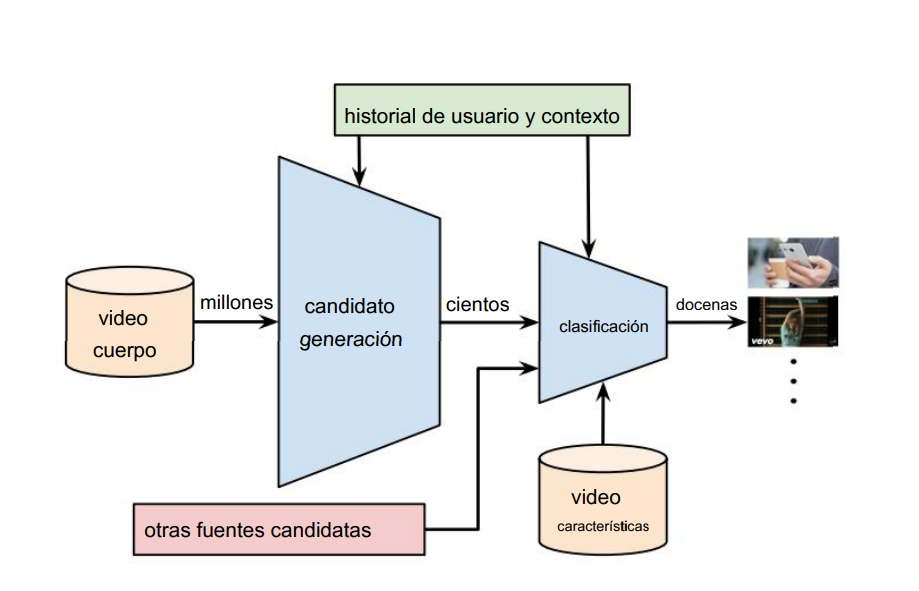


Figura 7 Arquitectura del Sistema de Recomendación de Youtube (Covington et al., 2016)

* **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, la factorización matricial (Steck et al., 2021).

Cada vez que se accede al servicio de Netflix, su sistema de recomendaciones 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:

* 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:

* 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).

* **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, 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.

El reranker, por su parte, es una herramienta sofisticada que predice las preferencias del usuario en múltiples dimensiones, lo que permite una clasificación más precisa y relevante de las aplicaciones sugeridas. Además, el optimizador de múltiples objetivos es una innovación clave que busca maximizar el valor de las métricas principales, como la relevancia y la satisfacción del usuario, mientras se mantienen en equilibrio las métricas secundarias, asegurando así una experiencia de usuario óptima.

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 en las recomendaciones, asegurando que las sugerencias sean justas y equitativas. Por último, el desarrollo de algoritmos capaces de encontrar equilibrios entre múltiples métricas asegura que el sistema no solo sea eficiente, sino también adaptable a las cambiantes necesidades y comportamientos de los usuarios (Gong & Zhernov, 2024).

**Ventajas y desventajas.**

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 y adoptar mejores prácticas, lo que permite establecer comparativas con los estándares de la industria. La innovación tecnológica es otro beneficio significativo, ya que el estudio de estos sistemas facilita la integración de tecnologías novedosas y algoritmos avanzados. Además, la personalización que caracteriza a estos sistemas puede ser emulada para ofrecer recomendaciones más precisas y atractivas a los usuarios.

Sin embargo, existen desventajas notables 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. Por último, el mantenimiento y actualización constantes representan un esfuerzo continuo y una inversión significativa.

### 1.2.2 Estudio de Sistemas Homólogos en el ámbito Nacional.

* **Sistema de recomendaciones sobre la evaluación de proyectos de desarrollo de software.**

En la actualidad existen organizaciones que asumen como estructura un modelo de gestión por proyectos. El creciente número de proyectos que se conceptualizan y se desarrollan, genera gran cúmulo de datos. Sin embargo, la insuficiente capacidad de análisis y procesamiento sobre los datos generados, imposibilitan la identificación de información que facilite la toma de decisiones a los principales directivos. Esta investigación propone el desarrollo de un Sistema de recomendación que integra la Sumarización Lingüística de Datos (LDS).

Este sistema de recomendación ayuda a los directores de proyectos a encontrar los indicadores a los que debe prestar especial atención por su impacto tanto positivo como negativo, ocultando aquellos indicadores que no le son útiles. Las explicaciones agregadas a las recomendaciones brindan la información necesaria para comprender por qué se recomendó determinado objeto de la base de datos. La utilización de un enfoque lingüístico borroso permitió transformar los datos numéricos en sentencias lingüísticamente interpretables para el usuario (Fonseca et al., 2021).

* **Sistema Recomendador de noticias para el portal Octavitos.**

En este proyecto se desarrolló un sistema recomendador de noticias, que mejoró la personalización de los contenidos presentes en el Portal Octavitos. Este sistema permite a los usuarios del Portal Octavitos satisfacer intereses informativos específicos, proporcionando un grupo de noticias recomendadas a partir de un perfil de usuario donde el mismo selecciona las temáticas que desea que se le recomiende y las evidencias de navegación dejadas en el portal.

En el desarrollo se utilizó Extreme Programming (XP) generándose los artefactos fundamentales que propone la metodología para cada etapa de trabajo, PHP como lenguaje de programación, las hojas de estilo en cascada (CSS) para la presentación de los contenidos y Drupal 7.15 como Sistema de Gestión de Contenido (CMS), logrando una integración eficiente (Cruz et al., 2013).

* **Sistema Recomendador para el turismo en Cuba.**

En el Caribe actual la competitividad en el mercado del turismo es muy fuerte, sin embargo, algunas empresas cubanas no acumulan tantos años de experiencia ni tienen presencia en tantos mercados internacionales. De ahí la necesidad de utilizar métodos efectivos para la toma de decisiones, vinculados a las Tecnologías de la Información y las Comunicaciones (TIC) y lograr la presencia de las ofertas de los destinos cubanos en todo el mundo.

En este trabajo se presenta un sistema capaz de dar al cliente una sugerencia hecha a partir de las características y preferencias del usuario, teniendo en cuenta las decisiones tomadas por los clientes anteriores con características similares. El sistema experto sugiere, según las características y preferencias, el destino turístico ideal para el cliente.

La tecnología empleada es el Razonamiento Basado en Casos (RBC) para desarrollar un sistema recomendador para el turismo cubano. Se analiza el problema del diseño de la base de conocimientos de manera que se alcance la rapidez necesaria en el sistema orientado a la WEB. Para dar solución a esta dificultad se propuso un sistema híbrido que combina RBC con árboles de decisión (Valdes Ramirez, 2013).

**Ventajas y desventajas**

Aunque estos sistemas nacionales no se utilizaron como guía en la propuesta de solución debido a sus diferencias tecnológicas, su estudio puede ofrecer lecciones sobre la implementación y optimización de sistemas de recomendaciones. Las ventajas incluyen la facilidad de implementación, la personalización y la relevancia de las recomendaciones, mientras que las desventajas abarcan la escalabilidad, la complejidad técnica y los requerimientos de mantenimiento. Estas consideraciones son importantes para el desarrollo de un sistema de recomendación basado en Deep Learning para las plataformas del proyecto z17, proporcionando un marco sólido para mejorar la eficiencia y efectividad del sistema propuesto.

### 1.2.3 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 comerciales internacionales estudiados suelen requerir licencias costosas y limitan la personalización 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.

La propuesta de solución busca desarrollar un sistema de recomendación que sea:

**Personalizable y dinámico:** permitiendo modificaciones y adaptaciones según las necesidades específicas de cada dominio.

**Alta precisión:** realiza recomendaciones eficientes y precisas de interés para los usuarios.

**Escalable:** capaz de manejar volúmenes de datos de hasta 50 y 100 millones de filas por plataforma, adaptándose a las crecientes demandas de las plataformas digitales Picta, todus y Apklis.

**Nacional:** un sistema soberano tecnológicamente del que no dependa de ninguna entidad externa para su uso.

La falta de sistemas de recomendación abiertos 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.

## 1.3 Propuesta de solución.

El sistema propuesto se inspira en el innovador estudio de Google *“Deep Neural Networks for YouTube Recommendations”* (Covington et al., 2016)*.* Este estudio 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 recomendaciónes que utiliza google en sus productos estrellas como youtube y Google play store (Covington et al., 2016). En las siguientes secciones, se ofrecerá una visión del funcionamiento de estos sistemas y como han transformado el campo de las recomendaciones personalizadas.

### 1.3.1 Recomendadores basados en Redes Neuronales Profundas.

**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. Estas herramientas avanzadas no solo 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 aprendizaje profundo, estos sistemas pueden procesar y cruzar una variedad de factores, desde el comportamiento de navegación en la web hasta las preferencias de compra y las interacciones sociales, para entregar recomendaciones altamente personalizadas que mejoran la experiencia del usuario** (Betru et al., 2017)**.**

**El aprendizaje profundo, que es el núcleo de estos sistemas, se inspira en la estructura y función del cerebro humano, particularmente en cómo las neuronas se conectan y transmiten información. Las redes neuronales artificiales están compuestas por capas de nodos interconectados 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 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)**.**

**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)**.**

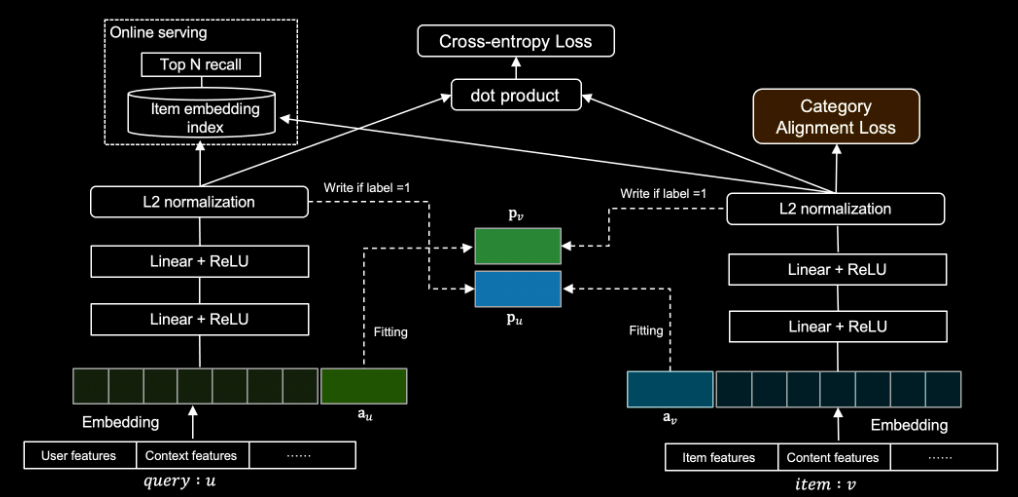


Figura 8 Arquitectura de una red neuronal profunda llamada Modelo de dos torres para recomendación (Kammoun et al., 2022).

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

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. La integración de un módulo de recomendaciones independiente en plataformas como Apklis, toDus y Picta puede transformar la experiencia del usuario al proporcionar sugerencias más relevantes y personalizadas.

El enfoque en la personalización y la velocidad de las recomendaciones es crucial, especialmente en un entorno donde el volumen de datos y la necesidad de actualización rápida son altos. Incorporar el análisis de metadatos del usuario y del contenido, así como el contexto temporal, permitirá que el sistema no solo recomiende contenido basado en la popularidad o similitudes superficiales, sino que también anticipe tendencias y responda a las preferencias cambiantes de los usuarios. Esto es particularmente importante en plataformas como Apklis, donde la personalización puede conducir a una mayor visibilidad de aplicaciones menos conocidas y potencialmente aumentar las ventas.

Además, la capacidad de procesar y aprender de las interacciones de los usuarios de forma ágil y rápida puede reducir significativamente los tiempos de cálculo, abordando el problema de los sistemas actuales que pueden tardar días en generar recomendaciones. Esto no solo mejorará la experiencia del usuario, sino que también hará que las plataformas sean más ágiles y capaces de adaptarse rápidamente a los cambios del mercado.

### 1.3.3 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, lo que puede llevar a un proceso de prueba y error que consume mucho tiempo. Otro desafío es garantizar que el sistema sea transparente y explicable, ya que las decisiones de recomendación deben ser comprensibles para los usuarios finales. La capacidad de explicar cómo y por qué se hizo una recomendación es crucial para la confianza y la aceptación del usuario (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 (Chen et al., 2023).

Pueden existir sesgos como:

* **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 (Chen et al., 2023).

Los sesgos se refieren a las tendencias o predisposiciones que se introducen en los sistemas de inteligencia artificial durante su desarrollo, entrenamiento o implementación, 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 (Chen et al., 2023).

Estos 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 hiperparametros de los modelos a utilizar.

La escalabilidad también es un desafío, ya que los sistemas de recomendación deben ser capaces de manejar un gran número de usuarios y productos sin degradar el rendimiento. Esto requiere una infraestructura robusta y eficiente.

## 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.

El proceso de desarrollo de software se apoya en el uso de diferentes herramientas y tecnologías, las cuales, unidas a la metodología seleccionada, conforman el ambiente de desarrollo de un sistema.

Una metodología de desarrollo de software se refiere al entorno que se usa para estructurar, planificar y controlar el proceso de desarrollo de un sistema informático. Tienen como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto, exigiéndose así que el proceso sea configurable. Las metodologías de desarrollo se clasifican en dos clases: las metodologías tradicionales o robustas y las ágiles o ligeras (González Matos, 2021).

**Metodología de desarrollo de software AUP versión UCI.**

La Universidad de las Ciencias Informáticas (UCI) desarrolló una versión de la metodología de desarrollo de software AUP (Proceso Ágil Unificado), con el fin de crear una metodología que se adapte al ciclo de vida definido por la actividad productiva de la universidad. Esta versión decide mantener para el ciclo de vida de los proyectos la fase de Inicio, pero modificando el objetivo de la misma y se unifican las restantes fases de la metodología de desarrollo de software AUP en una sola, nombrada Ejecución y agregándose también una nueva fase llamada Cierre (González Matos, 2021) (SÁNCHEZ, 2015).

Tabla 1 Fases de las variaciones de AUP para la UCI (González Matos, 2021).

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

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

**Escenario No 1:**

Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.



Figura 9 Metodología AUP versión UCI escenario 1.

**Escenario No 2:**

Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Captura de pantalla 2024-09-17 224237

Figura 10 Metodología AUP versión UCI escenario 2.

**Escenario No 3:**

Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Captura de pantalla 2024-09-17 224241

Figura 11 Metodología AUP versión UCI escenario 3.

**Escenario No 4:**

Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Captura de pantalla 2024-09-17 224245

Figura 12 Metodología AUP versión UCI escenario 4.

Teniendo en cuenta los escenarios de la variación AUP-UCI, se decide encapsular los requisitos en el escenario 2. Para la decisión se analizó el criterio del escenario que plantea que:

Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Se eligió principalmente este escenario ya que la propuesta de solución se caracteriza por ser un proyecto poco extenso a pesar de ser complejo lo cual es ideal para este tipo de escenario.

### 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 suelen usarse 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](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) [interpretado](https://es.wikipedia.org/wiki/Int%C3%A9rprete_(inform%C3%A1tica)) cuya filosofía hace hincapié en la legibilidad de su [código](https://es.wikipedia.org/wiki/Codigo_fuente). Se trata de un lenguaje de programación [multiparadigma](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n" \l "Paradigma_de_programaci%C3%B3n" \o "Lenguaje de programación), ya que soporta parcialmente la [orientación a objetos](https://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), programación imperativa y, en menor medida, [programación funcional](https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional). Es un [lenguaje interpretado](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_interpretado), [dinámico](https://es.wikipedia.org/wiki/Tipado_din%C3%A1mico) y [multiplataform](https://es.wikipedia.org/wiki/Multiplataforma)a («Python», 2024).

#### Ventajas de Python

Python ofrece una sintaxis clara y legible que facilita la lectura y escritura de código. Esto es particularmente beneficioso en proyectos donde se deben manejar grandes cantidades de datos y algoritmos complejos. 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. Esto significa que es fácil encontrar soluciones a problemas comunes y obtener ayuda cuando se necesita.

Python, con su ecosistema rico y en constante evolución, ofrece una plataforma sólida y flexible para desarrollar sistemas de recomendación. Su facilidad de uso, amplia gama de bibliotecas y capacidad de escalamiento lo convierten en la elección preferida para muchos científicos de datos e ingenieros de software (Alfaro & Roberto, 2022).

### 1.4.3 Sistema gestor de base de datos.

#### PostgreSQL V 15.2

PostgreSQL es un poderoso sistema de base de datos relacional de objetos de código abierto que usa y extiende el lenguaje SQL combinado con muchas características que almacenan y escalan de manera segura las cargas de trabajo de datos más complicadas. PostgreSQL se ha ganado una sólida reputación por su arquitectura probada, confiabilidad, integridad de datos, conjunto de características robustas, extensibilidad y la dedicación de la comunidad de código abierto (PostgreSQL Global Development Group, 2024).

### 1.4.4 Bibliotecas.

En [informática](https://es.wikipedia.org/wiki/Inform%C3%A1tica), una biblioteca o, llamada por vicio del lenguaje, librería es un conjunto de implementaciones funcionales, codificadas en un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n), que ofrece una interfaz bien definida para la funcionalidad que se invoca. A continuación, se describen las librerías o bibliotecas a utilizar en el desarrollo de la propuesta de solución (Tsui et al., 2022):

#### Tensorflow V 2.15.0

TensorFlow es una biblioteca de [código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto) para [aprendizaje automático](https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico) a través de un rango de tareas, y desarrollado por [Google](https://es.wikipedia.org/wiki/Google) para satisfacer sus necesidades de sistemas capaces de construir y entrenar [redes neuronales](https://es.wikipedia.org/wiki/Red_neuronal_artificial) para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

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 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 recomendación a gran escala.

Además cuenta con un ecosistema completo de herramientas específicamente para desarrollar sistemas de recomendaciones potentes (*TensorFlow*, s. f.).

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.

#### Keras V 2.15.0

Keras es una biblioteca de [Redes Neuronales](https://es.wikipedia.org/wiki/Red_neuronal_artificial) de [Código abierto](https://es.wikipedia.org/wiki/C%C3%B3digo_abierto) escrita en [Python](https://es.wikipedia.org/wiki/Python" \o "Python). Es capaz de ejecutarse sobre [TensorFlow](https://es.wikipedia.org/wiki/TensorFlow" \o "TensorFlow), [Microsoft Cognitive Toolkit](https://es.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit) o [Theano](https://es.wikipedia.org/w/index.php?title=Theano&action=edit&redlink=1" \o "Theano (aún no redactado)).

Está especialmente diseñada para posibilitar la experimentación en más o menos poco tiempo con redes de [Aprendizaje profundo](https://es.wikipedia.org/wiki/Aprendizaje_profundo). Sus fuertes se centran en ser amigable para el usuario, modular y extensible.

Keras se encarga de gestionar muchos de los detalles de implementación de bajo nivel, como la optimización, la propagación hacia atrás y la gestión de los tensores. Esto permitirá construir modelos de manera más rápida y con menos errores («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.  
  
TFRS posibilita lo siguiente:

* Compilar y evaluar modelos de recuperación de recomendación.
* Incorporar libremente [información de contexto](https://www.tensorflow.org/recommenders/examples/featurization?hl=es-419), artículos y usuarios en los modelos de recomendación.
* Entrenar [modelos para varias tareas](https://www.tensorflow.org/recommenders/examples/multitask/?hl=es-419) que permitan optimizar varios objetivos de recomendación en conjunto (*TensorFlow Recommenders*, s. f.).

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](https://es.wikipedia.org/wiki/Python) que da soporte para crear [vectores](https://es.wikipedia.org/wiki/Vector_(inform%C3%A1tica)) y [matrices](https://es.wikipedia.org/wiki/Vector_(inform%C3%A1tica)) grandes multidimensionales, junto con una gran colección de [funciones](https://es.wikipedia.org/wiki/Funci%C3%B3n_matem%C3%A1tica) [matemáticas](https://es.wikipedia.org/wiki/Matem%C3%A1ticas) de [alto nivel](https://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel) para operar con ellas (*NumPy -*, s. f.).

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](https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)) de Python especializada en la manipulación y el [análisis de datos](https://es.wikipedia.org/wiki/An%C3%A1lisis_de_datos). Ofrece estructuras de datos y operaciones para manipular tablas numéricas y [series temporales](https://es.wikipedia.org/wiki/Series_temporales), es como el Excel de Python. Es un [software libre](https://es.wikipedia.org/wiki/Software_libre) distribuido bajo la [licencia BSD](https://es.wikipedia.org/wiki/Licencia_BSD) (Berkeley Software Distribution). ​ El nombre deriva del término "[datos de panel](https://es.wikipedia.org/wiki/Datos_de_panel)", término de [econometría](https://es.wikipedia.org/wiki/Econometr%C3%ADa) que designa datos que combinan una dimensión temporal con otra dimensión transversal (*pandas - Python Data Analysis Library*, s. f.).

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 APIs con Python, que se basa en las pistas de tipo estándar de Python. Sus características clave incluyen un rendimiento muy alto, comparable con NodeJS y Go, gracias a Starlette y Pydantic, lo que lo convierte en uno de los frameworks de Python más rápidos disponibles.

Además, FastAPI facilita la codificación, aumentando la velocidad de desarrollo de características en un 200% a 300%, y reduce los errores inducidos por los desarrolladores en un 40%. Es intuitivo, con un excelente soporte de editor y autocompletado en todas partes, lo que permite menos tiempo dedicado a la depuración. Su diseño fácil de usar y aprender minimiza el tiempo dedicado a leer documentación (*FastAPI*, s. f.).

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, 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.

## 1.5 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 dominio de la presente investigación y las relaciones entre ellos. Esto permitió alcanzar un mayor dominio sobre los fundamentos de los Sistemas de Recomendaciones 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.
* El análisis del caso de estudio: Sistema de recomendación basado en deep learning para las plataformas Picta, Todus y Apklis las cuales pertenecen al proyecto Z17, proporcionó una oportunidad única para identificar con precisión las características distintivas de la propuesta de solución. Estas características son: Robustez, Redes neuronales profundas 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, ha proporcionado un entorno de desarrollo ágil y eficiente para la investigación y experimentación con diferentes algoritmos de recomendación. La integración de estas herramientas con PostgreSQL y el framerwork web Fastapi ha permitido 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.

**Introducción**

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.

Clasificación

Recuperación

Re-Clasificación

El sistema está compuesto por **3 etapas** fundamentales, **Recuperación, Clasificación y Re-clasificación**. Estas etapas están conectadas entre sí de manera secuencias respectivamente donde la salida de una es la entrada de la otra siendo la última etapa 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:

* **Recuperación**: Esta etapa es responsable de seleccionar un conjunto inicial de cientos o miles de candidatos entre todos los posibles candidatos 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 puede estar compuesta por uno o varios modelos generadores de candidatos donde cada uno se basa en los datos implícitos (véase epígrafe 1.1.2) almacenados de los usuarios (Gao et al., 2023).
* **Clasificación**: Toma los resultados del modelo de recuperación 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 (Gao et al., 2023).
* **Re-clasificación:** En la etapa final el sistema puede volver a clasificar para considerar criterios o limitaciones adicionales como pueden ser la actualidad, la diversidad y la equidad. Estos son algunos de los diversos factores que pueden ayudar a mejorar considerablemente las recomendaciones de un sistema.

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

**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 elimina la visibilidad del contenido, algo fundamental en un sistema de recomendación.

**La equidad** evita sesgos en las recomendaciones como podrían ser sebos de clics o falsas tendencias y permite que todos los usuarios sean tratados de manera justa.

Cada una de estas etapas está compuesta por 1 o varios modelos de redes neuronales profundas dependiendo de qué tan complejos sean los datos a utilizar (Covington et al., 2016; Gao et al., 2023).

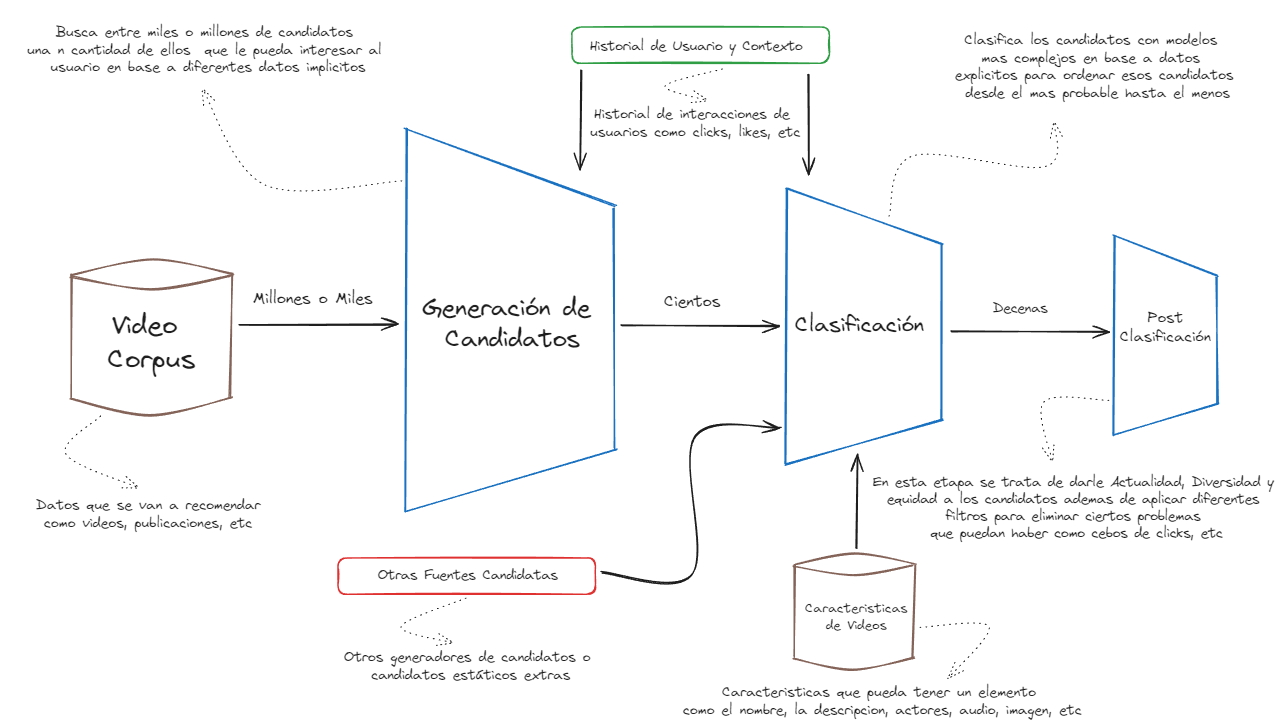


Figura 13 Arquitectura de capas del sistema.

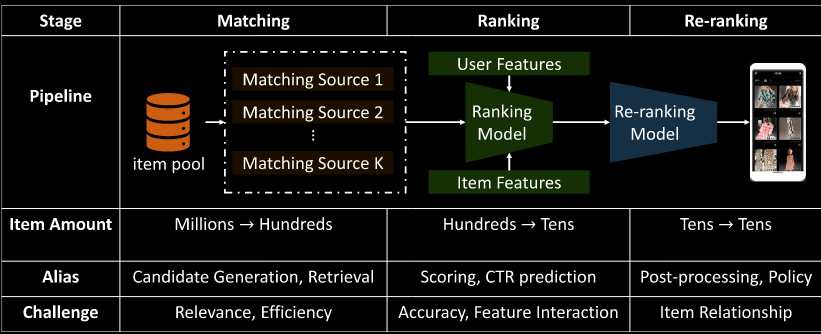


Figura 14 Estructura típica de un sistema de recomendación (Gao et al., 2023).

### 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.

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, podemos 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).

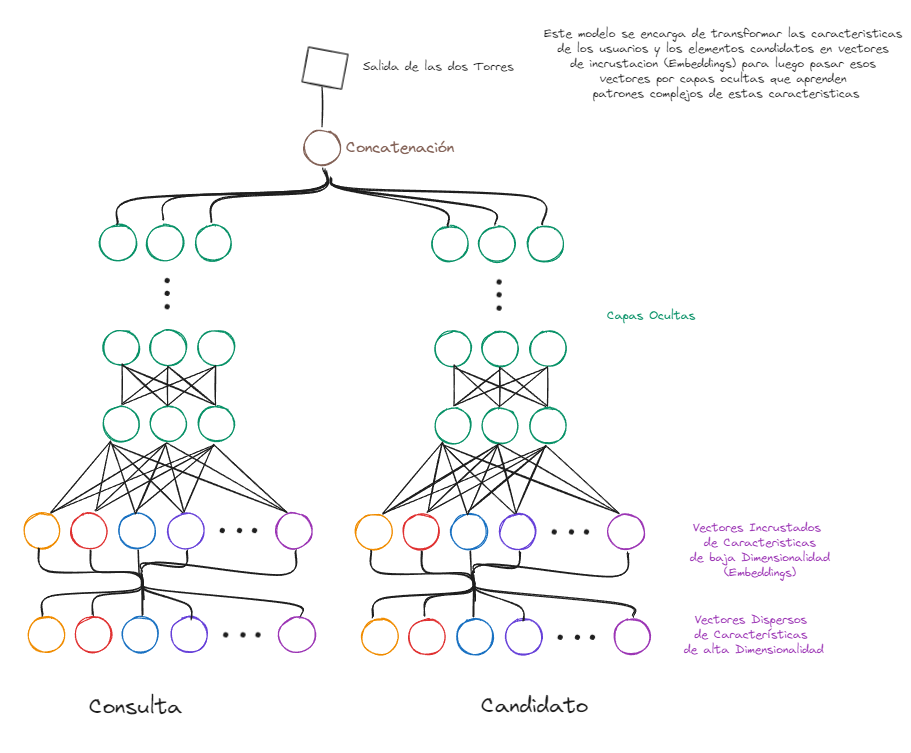


Figura 15 Arquitectura de red neuronal de dos torres.

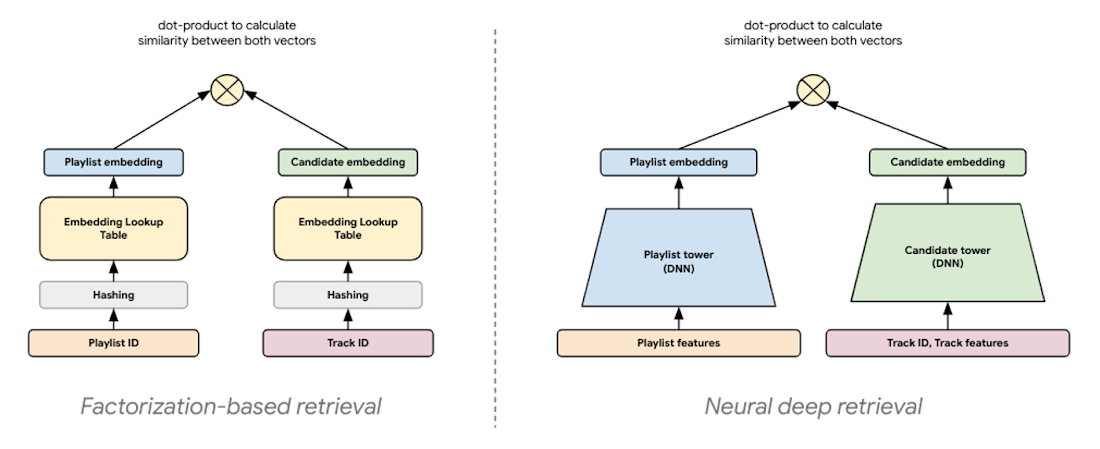


Figura 16 Las arquitecturas NDR (neural deep retrieval), como los codificadores de dos torres, son conceptualmente similares a los modelos de factorización.

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).

**Modelo de Recuperación**: Es el encargado de leer el corpus de datos completo y recuperar unos miles de 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.

* **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 que calcula la probabilidad de interés de un usuario hacia cada candidato (Covington et al., 2016). A continuación la función softmax.

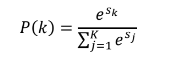


Figura 17 Función de activación softmax (Tomás Cruz, 2024).

Donde:

P(k) es la probabilidad de que la entrada pertenezca a la categoría "k".

sk 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** específica diseñada para evaluar el desempeño de modelos de recomendación factorizados. Esta métrica se enfoca en medir la capacidad del modelo para predecir correctamente los elementos (por ejemplo, productos, películas) 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, 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.

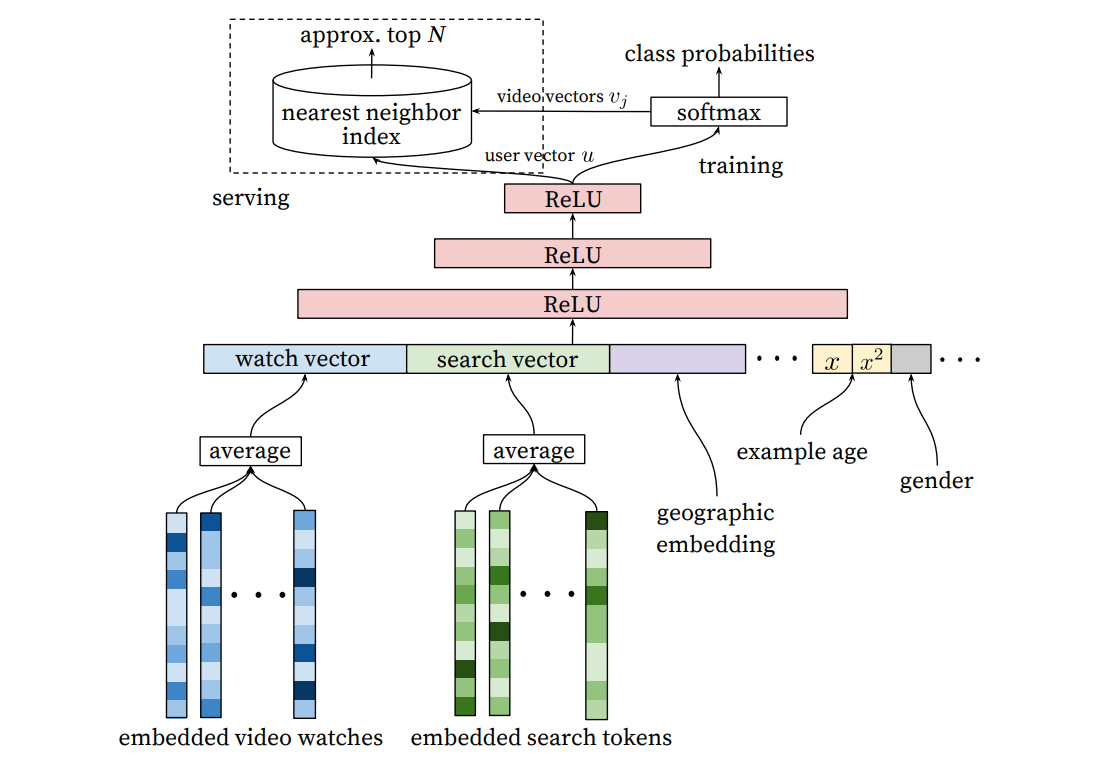


Figura 18 Arquitectura del modelo de recuperación (Covington et al., 2016)

**Modelo de Clasificación:** Después de la generación del candidato, otro modelo califica y clasifica los candidatos para seleccionar el conjunto de elementos que se mostrarán.

El sistema combina estas diferentes fuentes en un grupo común de candidatos que se califican con un solo modelo y se clasifican según esa puntuación. Por ejemplo, el sistema puede entrenar un modelo para que prediga probabilidad de que un usuario mire un video en YouTube a partir de los datos explícitos del mismo como podrían ser los likes y teniendo en cuenta los siguientes factores (Covington et al., 2016):

Funciones de búsqueda (por ejemplo, historial de reproducciones del usuario, idioma, país y hora).

Elementos de video (por ejemplo, título, etiquetas, incorporación de video, duración).



Figura 19 Arquitectura de un modelo de clasificación (Covington et al., 2016)

## 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, 2011).

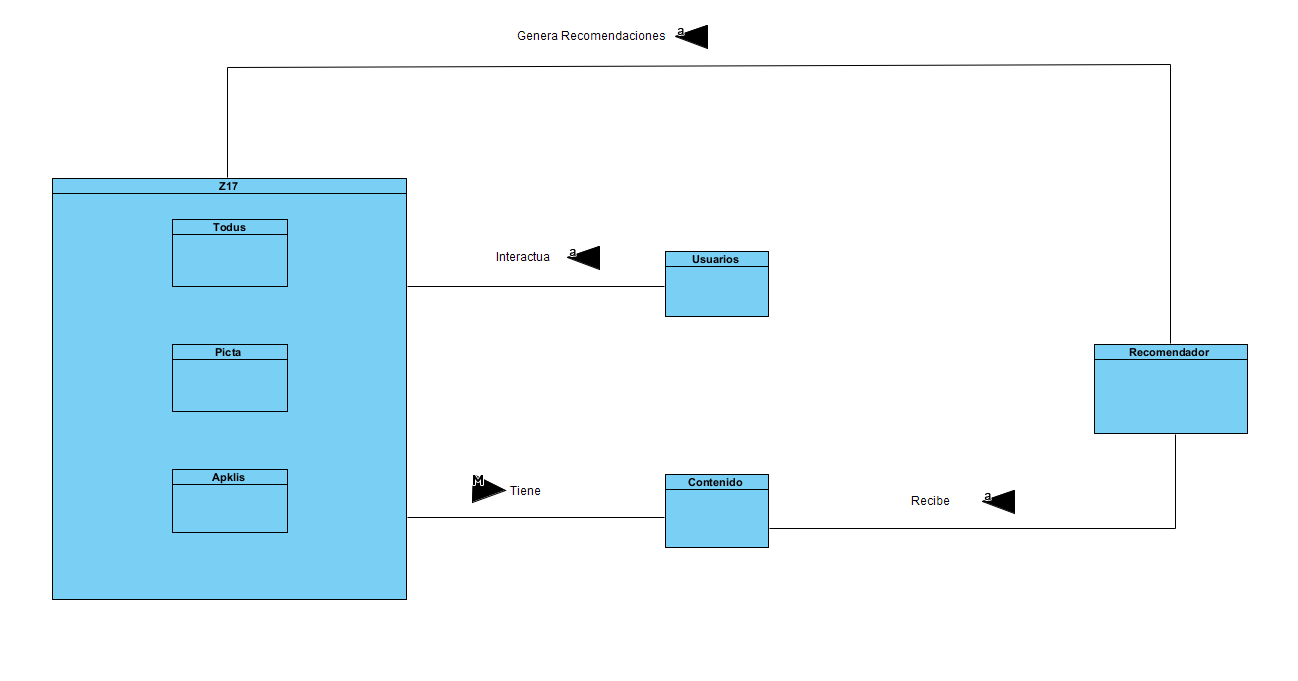


Figura 20 Modelo Conceptual

## 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) (Larman, 2003).

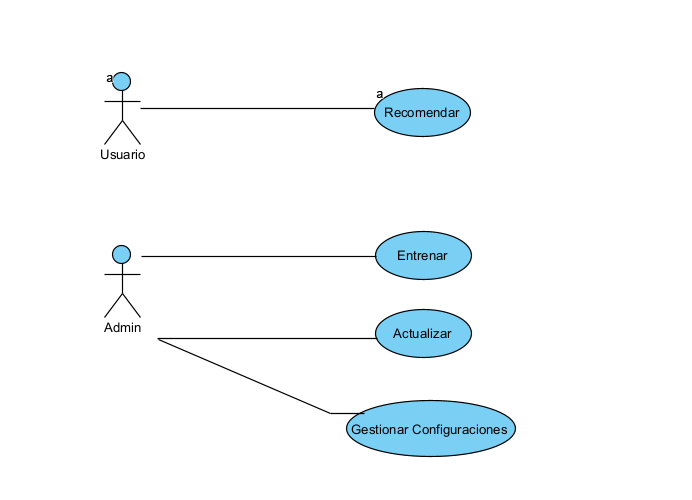


Figura 21 Diagrama de caso de uso del sistema.

## 2.4 Diagrama de clases de diseño.

Un diagrama de clases de diseño (DCD) representa las especificaciones de las clases e interfaces 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, 2011).

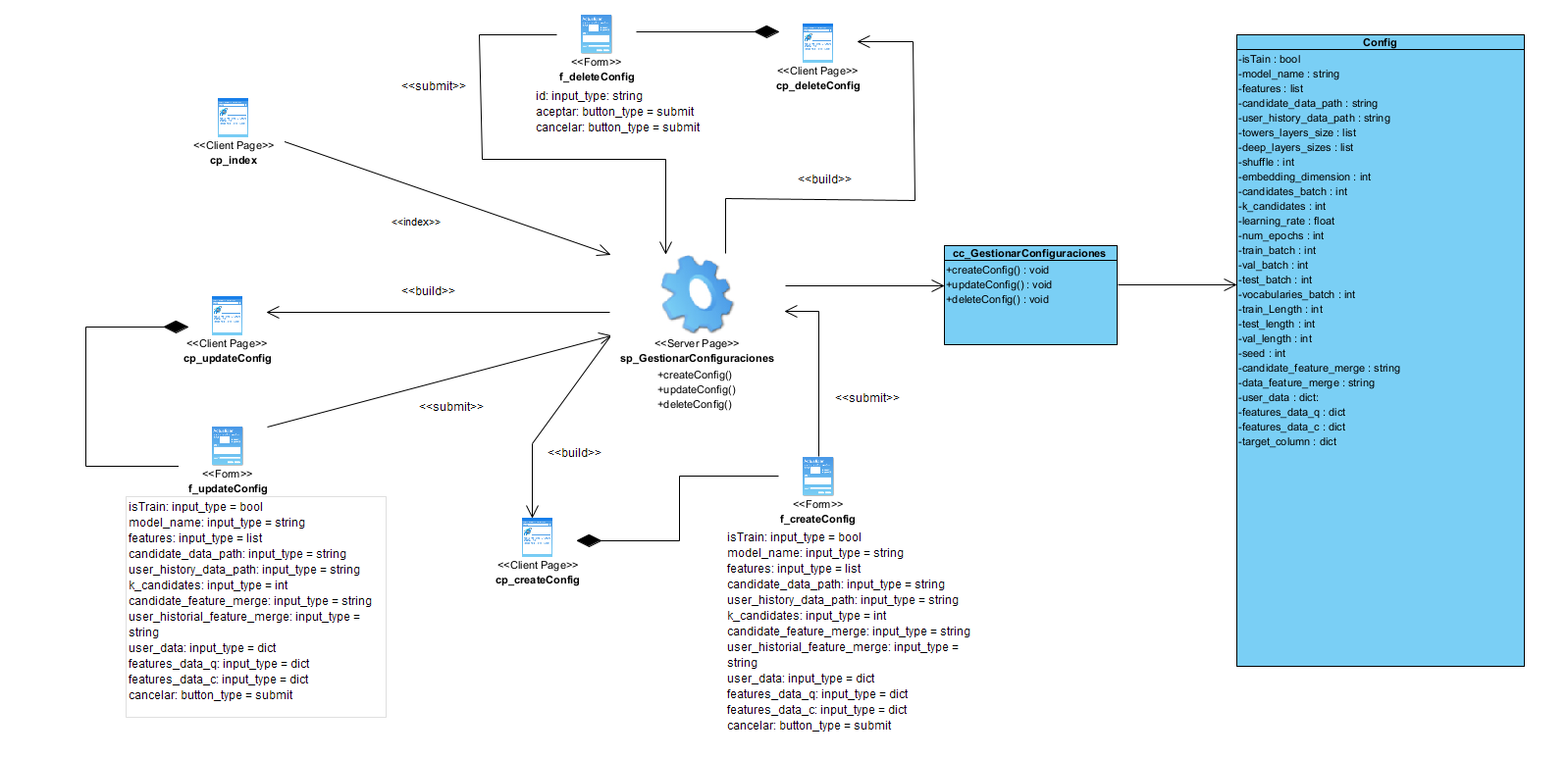


Figura 22 Diagrama de clases con estereotipos web.

## 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, 2011).

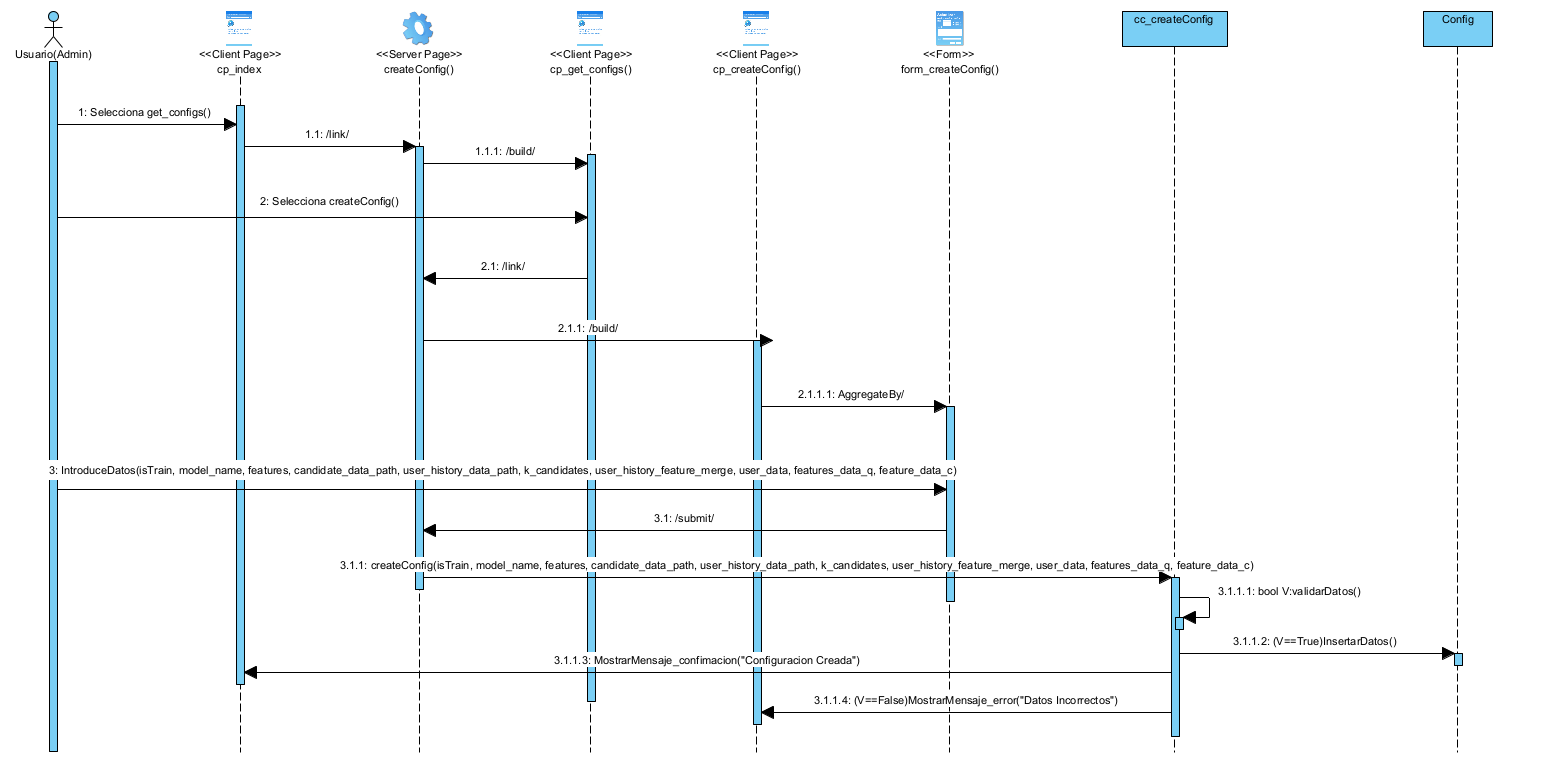


Figura 23 Diagrama de secuencia.

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 configuracion, 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 desde, 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 de la base, 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. Se basa en la identificación de los objetos primarios que va a procesar el sistema, la composición y atributos de los mismos. En algunos casos, esta base de datos es independiente del sistema software (González Matos, 2021) (Sommerville, 2011).

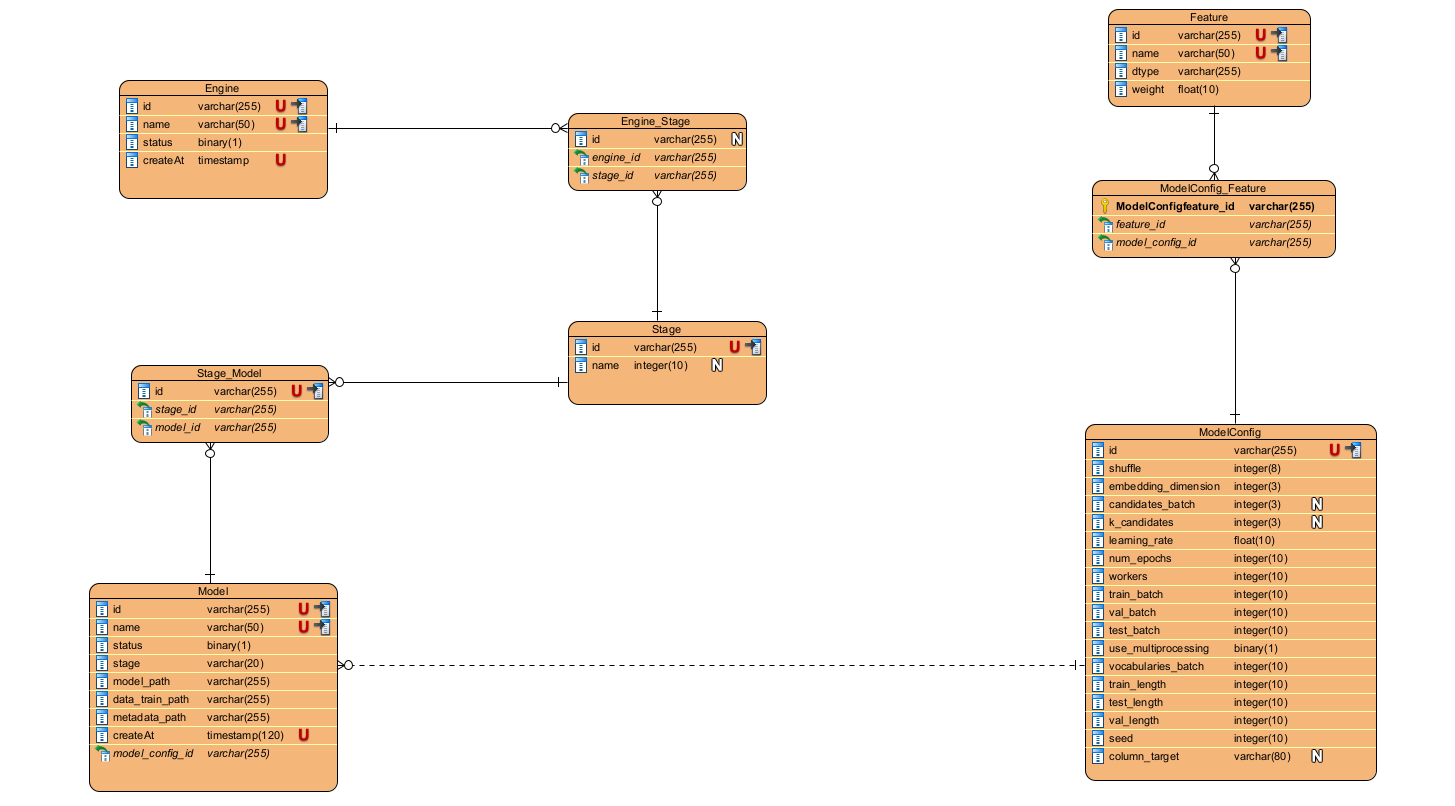


Figura 24 Modelo de datos.

## 2.7 Requisitos de la propuesta de solución

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 (González Matos, 2021) (Sommerville, 2011).

### 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 (González Matos, 2021) (Pressman, 2002).

Tabla 5 Descripción Requisitos Funcionales

|  |  |  |  |
| --- | --- | --- | --- |
| Requisitos Funcionales | Nombre del requisito funcional | Prioridad | Complejidad |
| RF1 | Entrenar | Alta | Alta |
| RF2 | Actualizar | Alta | Alta |
| RF3 | Recomendar | Alta | Media |
| RF4 | Crear Configuraciones | Alta | Media |
| RF5 | Actualizar Configuraciones | Media | Media |
| RF6 | Mostrar Configuraciones | Alta | Baja |
| RF7 | Listar Configuraciones | Baja | Baja |

### 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. Los requisitos no funcionales se aplican al sistema como un todo, más que a características o a servicios individuales (González Matos, 2021) (Sommerville, 2011).

Tabla 6 Descripción de los Requisitos no Funcionales

|  |  |
| --- | --- |
| 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. |
| Rendimiento | |
| RNF4 | El sistema debe permitir que los usuarios interactúen con él de manera concurrente. |
| RNF5 | El tiempo de demora de una petición al servidor debe ser menor de cinco (5) segundos aproximadamente. |
| Restricciones de Implementación y Diseño | |
| RNF6 | El sistema debe ser desarrollado en su totalidad con tecnologías de código abierto. |
| RNF7 | Servidor web ASGI Uvicorn. |
| RNF8 | Servidor de base de datos PostgreSQL v15 |
| RNF9 | Como lenguaje de programación se debe utilizar Python v3.11.9 |
| Software | |
| RNF10 | Para el uso del sistema se requiere una PC cliente con cualquier sistema operativo, que se pueda instalar navegadores web para el uso de la aplicación. |
| RNF11 | La comunicación entre la PC cliente y el servidor de aplicaciones web se realiza a través del protocolo HTTPS. |
| Hardware | |
| RNF12 | Teniendo en cuenta que este tipo de proyectos basados en redes neuronales profundas y análisis de grandes conjuntos de datos necesita de una muy buena potencia de cómputo para echar a andar de la cual no se dispone en el momento de implementación y prueba del mismo el autor ha establecido unos requisitos mínimos con los que cuenta para correr este sistema bajo ciertas condiciones.  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 para un modelo de x parámetros, un corpus de entrenamiento de 1 millón de filas y z columnas. |

## 2.8 Arquitectura de software

El sistema entero está formado por dos partes: un núcleo o engine el cual tiene una estructura basada en la explicación que dio 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 Apis del sistema con las cuales se pueden realizar diferentes acciones como entrenar el sistema, usarlo, actualizarlo, etc.

Este núcleo o engine 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 FastApi desarrollado en python el cual utiliza la arquitectura Modelo-Vista-Controlador (MVC).

La idea detrás de MVC es que cada uno de los componentes en su 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. Principalmente, tiene como objetivo dar soporte a los modelos funcionales y mapas mentales de la información relevante para los usuarios, permitiendo un modelo que facilite la consulta y manejo de los mismos. 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.

## 2.9 Patrones de Diseño

Los **patrones de diseño** son unas técnicas para resolver problemas comunes en el [desarrollo de software](https://es.wikipedia.org/wiki/Desarrollo_de_software) y otros ámbitos referentes al diseño de interacción o interfaces.

Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (González Matos, 2021) (Pressman, 2010).

### 2.9.1 Patrones GRASP

En [diseño orientado a objetos](https://es.wikipedia.org/wiki/Dise%C3%B1o_orientado_a_objetos), GRASP son *patrones generales de software para asignación de responsabilidades*, es el [acrónimo](https://es.wikipedia.org/wiki/Acr%C3%B3nimo) de "GRASP (object-oriented design General Responsibility Assignment Software Patterns)". Aunque se considera que más que [patrones](https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o) propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software (Botero Tabares, 2023).

#### Experto

El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos 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, cada modelo se encapsuló como una clase diferente guardando cada uno sus propias tareas y acciones.



Figura 25 Clase del modelo de recuperación donde se refleja el patrón Experto.

#### Creador

El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o [instanciación](https://es.wikipedia.org/wiki/Instancia_(programaci%C3%B3n))) de nuevos [objetos](https://es.wikipedia.org/wiki/Objetos_(programaci%C3%B3n_orientada_a_objetos)) o [clases](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)). 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 realizar funcionalidades del 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.



Figura 26 Acción Entrenar encargada de crear instancias de objetos

#### Alta cohesión

La información que almacena una [clase](https://es.wikipedia.org/wiki/Clase_(inform%C3%A1tica)) debe ser coherente y debe estar (en la medida de lo posible) relacionada con la clase (Botero Tabares, 2023).

#### Bajo acoplamiento

Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la [reutilización](https://es.wikipedia.org/wiki/Reutilizaci%C3%B3n_de_c%C3%B3digo), y disminuyendo la dependencia entre las clases (Botero Tabares, 2023).

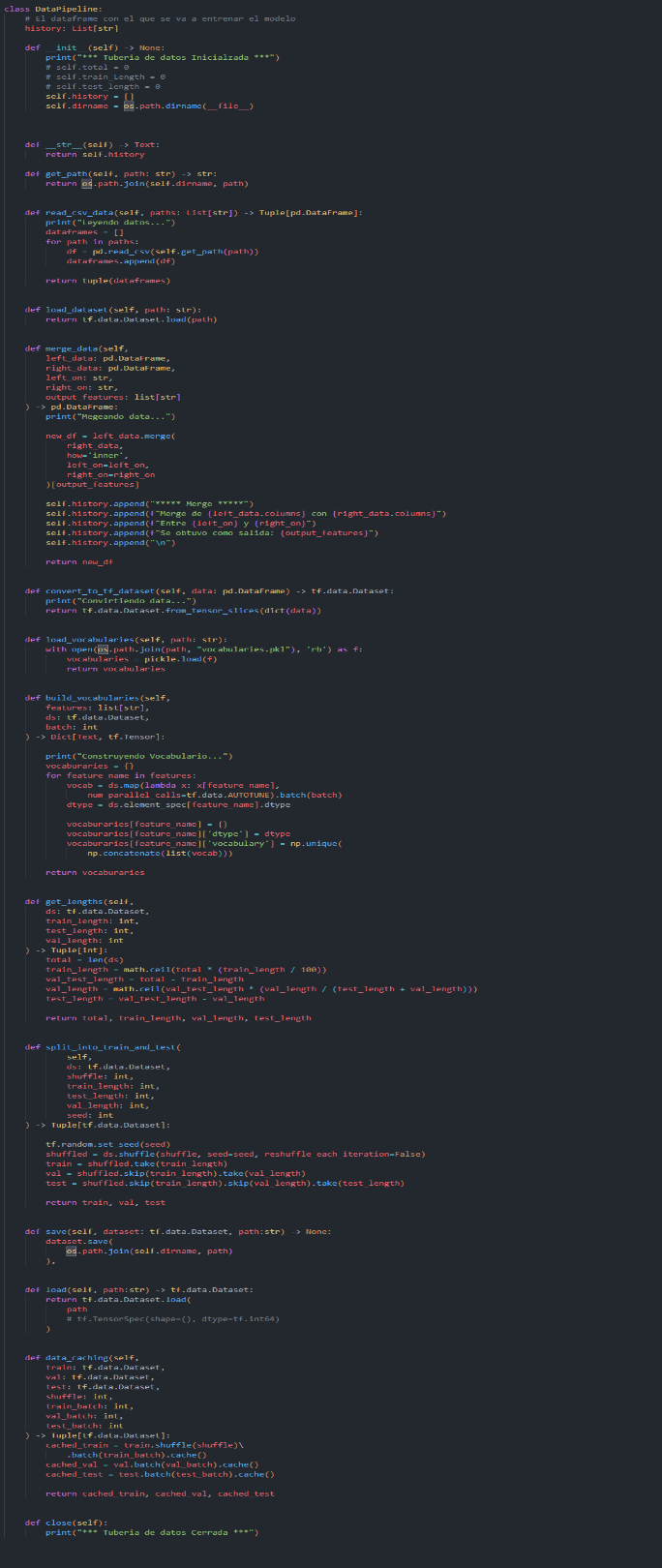
****

Figura 27 Clase DataPipeline utilizando el patrón alta cohesión.



Figura 28 Clase ModelConfig utilizando el patrón bajo acoplamiento.

### 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 (Guerrero et al., 2013).



Figura 29 Creación de dos instancias globales en el sistema.

#### Template Method

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

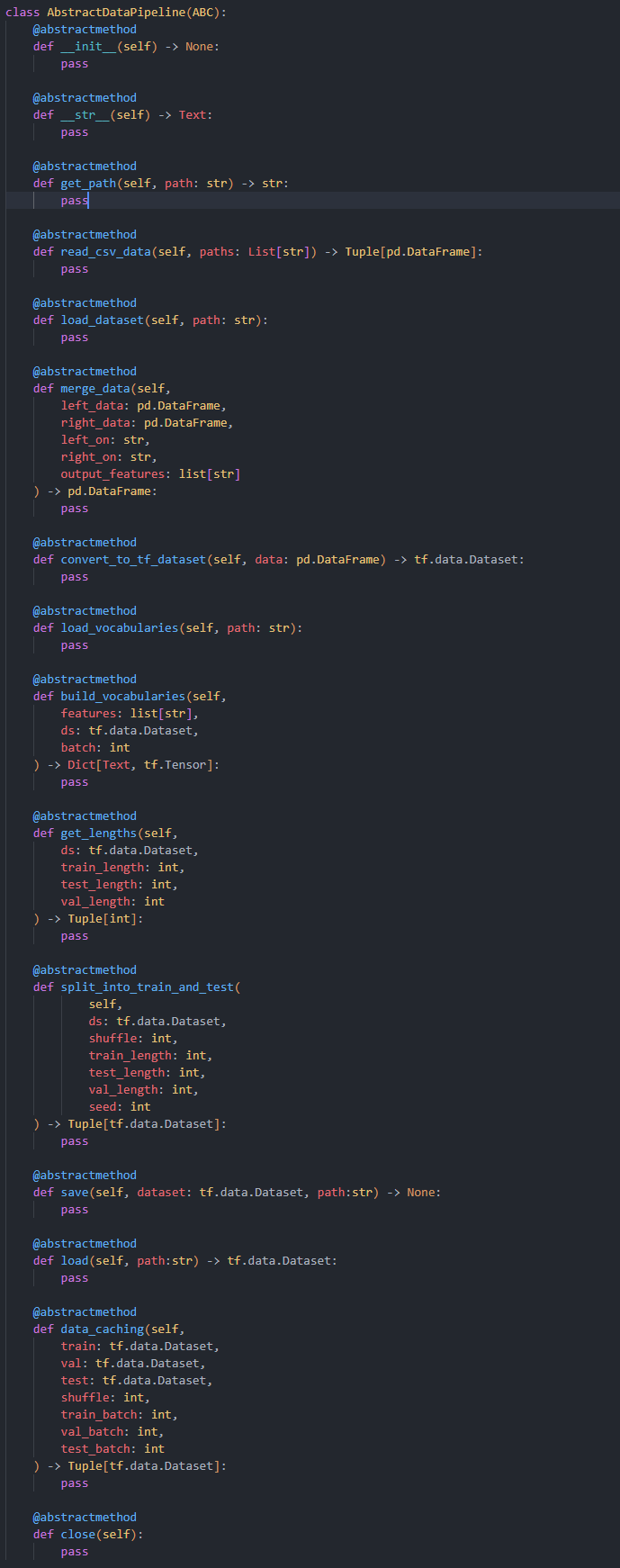
****

Figura 30 Clase interfaz que se utiliza como platilla para la clase DataPipeline.

## 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 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.

# Referencias Bilbiográficas

1. Acosta, G. (2020, mayo 21). Modelos de recomendación: Recomendando qué recomendar. *Quanam*. https://quanam.com/modelos-de-recomendacion-recomendando-que-recomendar/
2. 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
3. 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
4. Almaraz Pérez, A. (2013, septiembre 25). *Sistemas de recomendación*. Universidad Autónoma Metropolitana. https://doi.org/10.24275/uami.q811kj77c
5. 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).
6. Botero Tabares, R. (2023). *Grasp Patterns and Anti-Patterns: An Object Oriented Approach from Logic Programming*. http://hdl.handle.net/10785/13571
7. 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
8. 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
9. 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
10. Cruz, Y. P., Ramírez, F. M., Plasencia, I. E. Y., & Dosagües, R. L. (2013). *SISTEMA RECOMENDADOR DE NOTICIAS PARA EL PORTAL OCTAVITOS*.
11. *FastAPI*. (s. f.). Recuperado 16 de octubre de 2024, de https://fastapi.tiangolo.com/
12. Fonseca, B. B., Cornelio, O. M., Pupo, I. P., Fonseca, B. B., Cornelio, O. M., & Pupo, I. P. (2021). Sistema de recomendaciones sobre la evaluación de proyectos de desarrollo de software. *Revista Cubana de Informática Médica*, *13*(2). http://scielo.sld.cu/scielo.php?script=sci\_abstract&pid=S1684-18592021000200004&lng=es&nrm=iso&tlng=es
13. 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
14. 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/
15. 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
16. 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
17. 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
18. 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
19. Keras. (2024). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Keras&oldid=160878252
20. Microsoft. (2024). *Visual Studio Code—Code Editing. Redefined*. https://code.visualstudio.com/
21. *NumPy -*. (s. f.). Recuperado 4 de octubre de 2024, de https://numpy.org/
22. *pandas—Python Data Analysis Library*. (s. f.). Recuperado 10 de septiembre de 2024, de https://pandas.pydata.org/about/index.html
23. PostgreSQL Global Development Group. (2024, octubre 15). *PostgreSQL*. PostgreSQL. https://www.postgresql.org/
24. Python. (2024). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Python&oldid=161998593
25. 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
26. 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
27. 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
28. *TensorFlow*. (s. f.). Recuperado 4 de octubre de 2024, de https://www.tensorflow.org/
29. *TensorFlow Recommenders*. (s. f.). TensorFlow. Recuperado 10 de septiembre de 2024, de https://www.tensorflow.org/recommenders?hl=es-419
30. Tomás Cruz, A. (2024). *Reconocimiento de lenguaje mediante un clasificador multiclase SoftMax*. https://hdl.handle.net/20.500.12371/21553
31. Tsui, F., Karam, O., & Bernal, B. (2022). *Essentials of Software Engineering*. Jones & Bartlett Learning.
32. Valdes Ramirez, D. (2013, marzo). *(PDF) Sistema Recomendador para el turismo en Cuba*. Sistema Recomendador para el turismo en Cuba. https://www.researchgate.net/publication/275100060\_Sistema\_Recomendador\_para\_el\_turismo\_en\_Cuba
33. Varela, D., Aguilar, J., Montoya, E., & Monsalve-Pulido, J. (2020). *Propuesta Arquitectónica de un Sistema de Recomendación Híbrido Adaptativo*.
34. Xia, J. H. H. (2023). *DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA UNA EMPRESA DE SERVICIOS ONLINE*.
35. 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
36. *Z17*. (s. f.). Recuperado 9 de septiembre de 2024, de https://z17.cu/
37. Zhang, Q., Lu, J., & Zhang, G. (2021). *Recommender Systems in E-learning*. https://www.oaepublish.com/articles/jsegc.2020.06