Arquitectura de sistema recomendador híbrido para el matching entre pacientes y enfermeras

Caldera-Sánchez, C. J.[1] ✉; Nava-Sánchez, P. A.[2]

Datos de Adscripción:

[1] ✉ Christian Javier Caldera Sánchez. Director de tecnología en Narvalla Innovación S.A.P.I. de C.V. christian.caldera@outlook.com ORCID 0009-0000-2638-9737

[2] Pedro Antonio Nava Sánchez. Director general en Narvalla Innovación S.A.P.I. de C.V. ing.pedronava@gmail.com ORCID 0009-0001-6582-7928

Resumen **- Los sistemas recomendadores permiten analizar las preferencias de los usuarios para prever patrones de comportamiento, mejorando así su experiencia al usar el sistema. El objetivo de esta investigación es validar la viabilidad, tanto técnica como funcional, de la implementación de un sistema recomendador híbrido en la plataforma digital de la empresa Narvalla, empresa dedicada a ofrecer servicios de enfermería a domicilio mediante su aplicación. Se limita el alcance del documento a la fase de diseño del desarrollo de software, con énfasis en la arquitectura del ecosistema, sus entidades, componentes, herramientas y sus interacciones.**

Se utilizaron metodologías de desarrollo cuyo propósito es el de modelar la realidad de la empresa y sus procesos para plasmarlos en el núcleo del sistema, metodologías como: Event Storming, Domain-Driven Design, Clean Architecture y Agile. Estas filosofías permiten generar sistemas que evolucionan iterativamente, proveyendo flexibilidad y escalabilidad, características que se alinean con ecosistemas cloud-based, como el de Narvalla, el cual utiliza Microsoft Azure, por lo que se continúa con su uso para este proyecto. El sistema diseñado emplea técnicas de inteligencia artificial como lo son: Collaborative Filtering y Differential Evolution Optimization, además de técnicas como Cosine Similarity, Weighted Hybrid, N-Top y Root Mean Squared Error.

Como resultado de combinar una sólida visión de arquitectura, algoritmos inteligentes y cloud-computing, se logró diseñar un sistema que se integra al ecosistema de Narvalla, además de generar múltiples artefactos de arquitectura. Concluyendo así su viabilidad técnica, funcional y evolución a futuro.

Palabras Clave **- ARQUITECTURA DE SOFTWARE, COMPUTACIÓN EN LA NUBE, DOMAIN-DRIVEN DESIGN, FILTROS COLABORATIVOS, MACHINE LEARNING, SISTEMA RECOMENDADOR HÍBRIDO.**

Abstract **- Recommender systems allow the analysis of user preferences to foresee behavior patterns, thus improving their experience when using the system. The objective of this research is to validate the feasibility, both technical and functional, of implementing a hybrid recommender system on the digital platform of the company Narvalla, a company dedicated to offering home nursing services through its application. The scope of this document is limited to the software development design phase, with emphasis on the architecture of the ecosystem, its entities, components, tools, and their interactions.**

Development methodologies were used whose purpose is to model the reality of the company and its processes to capture them in the core of the system, methodologies such as: Event Storming, Domain-Driven Design, Clean Architecture, and Agile. These philosophies allow the generation of systems that evolve iteratively, providing flexibility and scalability, characteristics that align with cloud-based ecosystems such as Narvalla's, which uses Microsoft Azure, and therefore its use continues for this project. The designed system employs artificial intelligence techniques such as Collaborative Filtering and Differential Evolution Optimization, in addition to techniques such as Cosine Similarity, Weighted Hybrid, N-Top, and Root Mean Squared Error.

As a result of combining a solid architectural vision, intelligent algorithms, and cloud computing, it was possible to design a system that integrates into Narvalla's ecosystem, in addition to generating multiple architecture artifacts. Thus, concluding its technical and functional feasibility and its future evolution.

Keywords **- COLLABORATIVE FILTERS, CLOUD COMPUTING, DOMAIN-DRIVEN DESIGN, HYBRID RECOMMENDER SYSTEM, MACHINE LEARNING, SOFTWARE ARCHITECTURE.**

I. INTRODUCCIÓN

Los sistemas recomendadores (SR) brindan a las organizaciones soluciones de procesamiento y análisis de información referente a las preferencias de sus consumidores, estas preferencias son después utilizadas para predecir patrones de conducta que son aprovechados para mejorar la experiencia del usuario (Caldera Sánchez et al., 2015). A los SR que utilizan más de una técnica de inteligencia artificial (IA) (NASA, s. f.) se les denomina híbridos (Coralogix, 2023).

Narvalla Innovación es una empresa dedicada a ofrecer servicios de enfermería a domicilio mediante su plataforma online. La finalidad de este proyecto es aprovechar los beneficios de un SR para ofrecer resultados de búsquedas personalizadas y recomendaciones de enfermeras a los pacientes.

Existen distintas técnicas de algoritmos recomendadores, para este caso de estudio se utilizará "Collaborative Filtering: User-based" (Coralogix, 2023), cuya función es predecir calificaciones de usuarios hacia ítems (en este caso, las

enfermeras) que aún no han evaluado. Además, este algoritmo permite generar un perfil del cliente que será utilizado para calcular un "Score" (calificación) de los ítems para así tomar los "Top-N" (Winand, s. f.) para ser recomendados. Este ranking tomará en consideración varios atributos del ítem por lo que una tabla de decisiones será necesaria, esto se puede llevar a cabo mediante la técnica de "Weighted Hybrid" (Híbrido Ponderado) (Chiang, 2024). Finalmente, será necesario que estas ponderaciones (pesos) estén en constante evolución, por lo que un método de "Machine Learning" (IBM, 2025) será necesario.

El objetivo de este artículo es validar la viabilidad técnica del proyecto, mediante la documentación de la etapa de diseño de arquitectura de software (Huet, 2022) de un sistema recomendador híbrido, para su integración en el ecosistema digital de Narvalla. Para ello, se deberá analizar la viabilidad del proyecto desde una perspectiva técnica, considerando los componentes y herramientas requeridos.

Al ser una implementación personalizada (Beer, 2024), se busca una integración fácil y eficiente con la infraestructura actual de la empresa, de la cual destacan las siguientes tecnologías utilizadas: Angular.io (Angular, s. f.) y Bootstrap (Bootstrap, s. f.) para el Front-End (Amazon, s. f.); C# .NET (Pedamkar, 2023) y ASP.NET (Pedamkar, 2023) para el Back-End (Amazon, s. f.), Azure SQL (Microsoft, s. f.-a) y Azure Blob Storage (Microsoft, s. f.-a) para almacenamiento de información; y Microsoft Azure (Microsoft, s. f.-b) como plataforma de computación en la nube (Microsoft, s. f.-b).

II. PARTE TÉCNICA DEL ARTÍCULO

A. Técnicas y metodologías de arquitectura de software

La visión de arquitectura propuesta para este proyecto será ejecutada con un enfoque de arquitectura evolutiva (García, 2021), es decir, se buscará generar un desarrollo iterativo, incremental y flexible que permita la integración de nuevas funcionalidades al sistema sin incurrir en retrabajo.

Dado que el alcance de este documento es el de analizar los requerimientos establecidos por Narvalla y con ellos diseñar una arquitectura de software para su futura implementación, se ha seleccionado la metodología "Domain-Driven Design", la cual es una técnica de desarrollo de software que enfatiza la comprensión del dominio del negocio (Neto, 2023), en otras palabras se busca garantizar el éxito del proyecto mediante una arquitectura que refleje con precisión la realidad del negocio, sus entidades, interacciones, procesos y responsabilidades de cada componente. Otro elemento de interés del DDD son los "Domain Events" (Neto, 2023), los cuales son eventos generados al realizar cambios de estado en las entidades, mensajes que serán después procesados por otras partes del sistema, desacoplando dependencias y permitiendo con ello cambios incrementales en la aplicación.

Se propone, además, el uso de sesiones de "Event Storming" (Neto, 2023) las cuales ayudaran a detectar procesos, entidades y domain events para esta y futuras iteraciones del desarrollo del proyecto.

Se continuará el uso de "Clean Architecture" (Martin, 2012) y C# utilizado actualmente en Narvalla para el diseño de cada componente de software, esta filosofía también permite un crecimiento incremental, y además se alinea con el uso de DDD, ya que enfoca sus esfuerzos en tener el dominio en el núcleo de la aplicación y modelar sus procesos apegados a la realidad de las empresas.

Finalmente, y aunque la etapa de programación no está contemplada como parte del alcance de este documento, se propone el uso de una metodología de desarrollo iterativo como lo es Agile (Agile Alliance, 2024).

B. Requerimientos

El primer paso en el proceso de arquitectura de software es la revisión de los requerimientos del proyecto, estos guiarán la toma de decisiones técnicas, las herramientas necesarias, los algoritmos a implementar y la interacción entre cada uno de los componentes.

1) Objetivo general del proyecto de software

Como se mencionó anteriormente, el objetivo de este artículo es definir la viabilidad del proyecto. Para ello, es necesario conocer el objetivo del proyecto en sí, mismo que se indica a continuación:

Integrar los beneficios de un SR en el ecosistema tecnológico de Narvalla, buscando así mejorar la experiencia de los pacientes mediante recomendaciones personalizadas basadas en sus preferencias y transacciones.

Para poder lograr este objetivo general del proyecto, se han definido siete requisitos que a su vez fungen como sub-objetivos, a continuación, se describen de manera sintetizada estos requerimientos recopilados mediante una sesión de "Event Storming".

2) REQ01: Fácil integración

El SR deberá integrarse eficientemente con la infraestructura actual de Narvalla, permitiendo así su personalización actual y a futuro.

3) REQ02: Calificaciones y atributos

Los pacientes podrán calificar varios atributos de las enfermeras(os) después de cada servicio efectuado. Los atributos para calificar serán: dominio técnico, empatía y trato humano, puntualidad, comunicación y, organización y limpieza. Estos atributos se calificarán del 1 al 5 en incrementos de 0.5 representados por estrellas y medias estrellas, siendo 5 una calificación perfecta.

4) REQ03: Recomendaciones desde el inicio

Los pacientes podrán obtener los beneficios del sistema recomendador tan pronto como sea posible sin importar que aún no cuenten con transacciones en el sistema.

5) REQ04: Búsquedas basadas en recomendaciones

Al realizar una búsqueda de enfermeras y sin importar los filtros utilizados, se mostrarán por default de mayor a menor las enfermeras según su probabilidad de ser contratadas por el paciente.

6) REQ05: Mejora del modelo de recomendación

El SR deberá mejorar sus recomendaciones con cada calificación que el cliente realice dentro del sistema.

7) REQ06: Recomendaciones multipropósito

Los beneficios del SR podrán ser usados en diversas áreas de la plataforma y no solo en los resultados de las búsquedas, por ejemplo, en la sección de "Servicio de curación de heridas", el paciente podrá ver una lista de las enfermeras que más se apegan a sus preferencias y que ofrezcan ese servicio en particular.

C. Estado actual de la aplicación

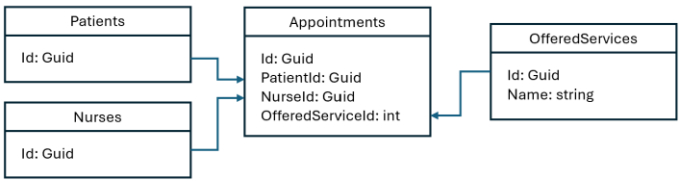
Antes de continuar con el diseño del SR, se evaluará primero el estado actual de la aplicación, sus entidades, componentes e infraestructura, lo cual servirá como punto de partida. Se omitirán detalles innecesarios como propiedades, componentes o aplicaciones que no estén relacionados al subdominio del SR, esto con la finalidad de proteger la propiedad intelectual de Narvalla.

1) Entidades

En la Figura 1 se muestra el diagrama entidad-relación (Lucidchart, s. f.) del subdominio de agenda de citas, en el cual se aprecian 4 entidades: Appointments (citas), Patients (pacientes), Nurse (enfermeras) y OfferedServices (servicios ofrecidos).

Figura 1

Entidades y relaciones actuales del subdominio de Agenda de Citas.

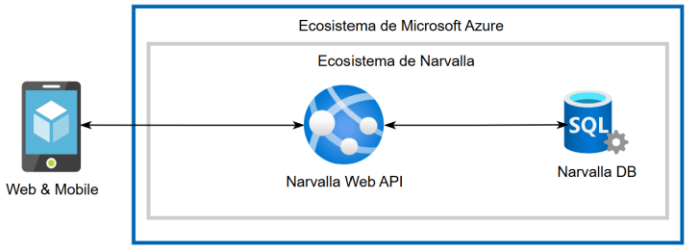


2) Infraestructura

En la Figura 2, es posible apreciar la infraestructura actual de la aplicación. El Servicio Web utiliza C# .NET y ASP.NET Web API (Postman, s. f.) como herramientas de desarrollo, cuenta con diversos endpoints REST (Postman, s. f.) para su conectividad con diferentes tipos de clientes, está hospedada en Azure App Service (Microsoft, s. f.-a), y la base de datos está hospedada en Azure SQL (Microsoft, s. f.-a).

Figura 2

Infraestructura actual de la plataforma.



D. Arquitectura

Una vez revisados los requerimientos y el estado actual de la aplicación, se procederá a definir la arquitectura de la plataforma. Por cada requisito REQ## le empatará un ítem ARQ##, el cual definirá cómo resolverá el problema planteado en el requisito. Cabe mencionar que los términos usuario y paciente son intercambiables al igual que ítem y enfermera(o).

1) ARQ01: Integración con infraestructura actual

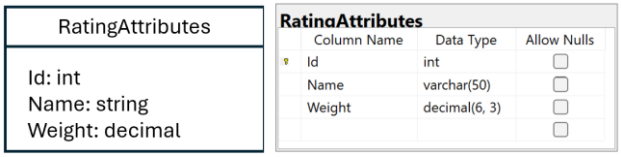
Se continuará utilizando el lenguaje de programación C# .NET para poder reutilizar cualquier biblioteca (ByteHide, 2024) interna o pública dentro de las aplicaciones a desarrollar. Se continuará utilizando la misma base de datos en Azure SQL para poder realizar filtros, uniones y reportes de forma eficiente dentro de la aplicación, además asegurará así la integridad de los datos, sin embargo, se creará un nuevo esquema (Coursera, 2025) en la base de datos (BD o DB en inglés) (Oracle, 2020) llamado "recommendations", para agregar todo lo referente al dominio del SR. Se continuará haciendo uso de la plataforma de Azure para poder utilizar cualquier otro servicio que permita la interacción entre componentes distribuidos dentro de la misma nube.

2) ARQ02: Atributos de evaluación

Se definirá una nueva entidad en el sistema llamada RatingAttributes (Atributos de Evaluación) la cual tendrá una tabla con el mismo nombre en la DB, véase la Figura 3, y en ella se almacenarán todos los atributos que el SR va a evaluar para generar los scores de recomendación.

Figura 3

Entidad RatingAttributes y su representación en la base de datos.



3) ARQ03: Solución al problema del inicio en frío

Uno de los problemas más comunes para un SR es el Cold Start (inicio en frío), sucede cuando un nuevo usuario se registra en el sistema y se tiene información limitada acerca de él y de sus preferencias. Esto dificulta proveer recomendaciones personalizadas (Culibrk, 2024).

Una de tantas técnicas para solucionar el problema del inicio en frío es el de solicitar explícitamente por retroalimentación (Tapereal, 2024), en la cual, se solicita información al usuario acerca de sus preferencias durante el proceso de onboarding al sistema (Tapereal, 2024). Para ello, se ha decidido el uso de un formulario que sea el último paso en el proceso del registro del usuario.

Este, le pedirá que ordene de mayor a menor los atributos de evaluación (mencionados en el REQ02) según la importancia que les da el usuario (véase la Figura 4). Al ordenarlos, se les asignará un valor de 5 al de más arriba, con decrementos de 0.5 a los otros atributos según su posición en la lista, esto permitirá crear un perfil rudimentario de las preferencias del usuario, véase el ejemplo en la Tabla 1.

Este perfil inicial se utilizará para registrar una calificación a una "Enfermera Cero", la cual, será virtual, es decir, no será una enfermera real, pero permitirá ingresar una transacción inicial a la plataforma para este usuario, véase la Figura 5 inciso a.

Tabla 1
Perfil rudimentario de preferencias de usuario.

	Dominio técnico	*Empatía y trato humano*	*Puntualidad*	*Comunicación*	*Organización y limpieza*
Nuevo usuario	5	3	3.5	4.5	4

Figura 4
Formulario de generación de perfil rudimentario de usuario y enfermera.

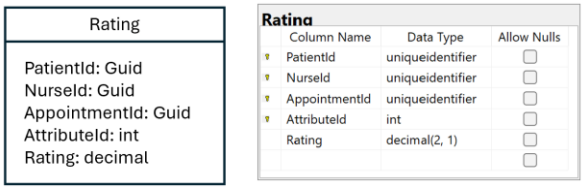


Figura 5
Tipos de calificaciones entre usuarios y enfermeras.



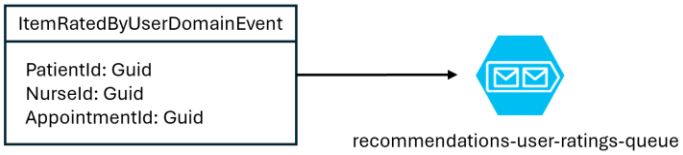
De manera similar, se le pedirá a la enfermera al registrarse, que ordene sus atributos tal y como se muestra en la Figura 4, sin embargo, la enfermera no tendrá un perfil de calificación, ya que esa entidad no evalúa servicios, en su lugar, se creará un usuario del sistema llamado "Usuario Cero", el cual tomará esos atributos y generará un registro de calificación para una "Cita Cero", tal y como se muestra en la Figura 5 inciso b. Esta y todas las calificaciones serán representadas por la entidad Rating (calificación) y almacenadas en la tabla de la DB con el mismo nombre y atributos; véase la Figura 6. En cuanto a los pacientes y enfermeras que ya no son nuevos en el sistema, se les mostrará la misma solicitud al ingresar a su cuenta para recopilar su información.

Figura 6
Entidad Rating y su definición en la base de datos.



Finalmente, cada vez que se ejecute una calificación por parte de un usuario a un ítem, se generará un Domain Event llamado "ItemRatedByUserDomainEvent", el mensaje deberá contener las propiedades mostradas en la Figura 7, este evento deberá ser enviado a una cola de mensajes en Azure Queue Storage (Microsoft, s. f.-a) llamada "recommendations-user-ratings-queue" para su futuro procesamiento, el cual está definido en la sección ARQ05.

Figura 7
Domain event ItemRatedByUserDomainEvent y su respectivo queue.



Todas estas acciones nos permitirán evitar un inicio en frío, ya que el usuario y el ítem, tendrán por lo menos un perfil inicial que se irá ajustando poco a poco a la realidad conforme se generen más transacciones en el sistema.

4) ARQ04: Sistema recomendador

Para el caso de Narvalla, se ha determinado que se utilizará un enfoque híbrido, es decir, que más de una técnica de IA será utilizada. La primera en utilizarse será la técnica de filtros colaborativos, los cuales generan recomendaciones basadas en la interacción y recolección de datos de otros usuarios dentro del sistema, esta técnica, aunque sencilla, es muy eficiente. Hay dos tipos de filtros colaborativos (FC), los basados en usuarios y los basados en ítems (Kurama, 2025), la diferencia radica en dónde se enfocará la atención de los resultados; dado que el enfoque de esta aplicación es mejorar la experiencia del usuario, se optará por el User-based.

a) Representación de vectores

Si pensamos en los usuarios como vectores compuestos por cada una de las calificaciones que han realizado en el sistema, podremos tener una representación matemática de ellos, lo cual permitirá realizar cálculos computacionales. Por ejemplo, observe la Tabla 2, en la que se muestra al usuario a ha evaluado los ítems 1, 3, 5 y 6, donde el valor de cada celda representa una calificación del usuario. Dado que en Narvalla se evalúan diferentes atributos, se tendrá entonces un vector por cada atributo del usuario.

Tabla 2
Calificaciones del usuario representadas por un vector.

	Ítem 1	*Ítem 2*	*Ítem 3*	*Ítem 4*	*Ítem 5*	*Ítem 6*
Usuario a	4.5		5		2.5	4

b) Similitud del coseno

Ahora que podemos definir a los usuarios como vectores, es posible calcular con ellos la similitud del coseno (Kurama, 2025), técnica que permite comparar 2 vectores para obtener su similitud. Esta similitud nos permitirá detectar usuarios similares al usuario a analizar y usarlos para predecir calificaciones faltantes en su vector. A continuación, se muestra la Tabla 3, en la cual se especifica el vector del usuario i y del usuario j para uno de los atributos evaluados, además de la fórmula mostrada en (1) para calcular su similitud:

Tabla 3

Vectores de usuario y el cálculo de su similitud

	Ítem 1	Ítem 2	Ítem 3	Ítem 4	Ítem 5	Ítem 6
Usuario a	4.5		5		2.5	4
Usuario b	4	3		5	3	

$$similitud(U_i, U_j) = \cos(\theta) = \frac{U_i \,.\, U_j}{|U_i| \ * |U_j|} \tag{1}$$

Mientras más cercana a 1, más similares son los vectores, y mientras más cercana a 0, menos similares son. Estas similitudes se deberán de realizar para cada par de usuarios en el sistema, creando así una matriz de similitudes, considerando que existirá una matriz para cada atributo almacenado en la tabla de RatingAttributes.

c) Predicción de calificaciones

Generar predicciones de calificaciones faltantes es importante a la hora de recomendar un ítem, ya que la idea principal del algoritmo es ofrecerle al usuario un ítem que creemos es altamente probable que se ajuste a sus preferencias, sin haberlo consumido anteriormente. En (2) se detalla la fórmula para predecir calificaciones faltantes (Ritvikmath, 2020), donde r_{U_1,I_a} es la predicción que el usuario 1 le dará al ítem a, $sim(U_1, U_x)$ es la similitud entre el usuario 1 y el usuario x, $sim(U_1, U_y)$ es la similitud entre el usuario 1 y el usuario y, C_{Ux,I_a} es la calificación que el usuario x le dio al ítem a y C_{Uy,I_a} es la calificación que el usuario y le dio al ítem a.

$$r_{U_1,I_a} = \frac{sim(U_1, U_x)C_{Ux,I_a} + sim(U_1, U_y)C_{UyI_a}}{sim(U_1, U_x) + sim(U_1, U_y)} \tag{2}$$

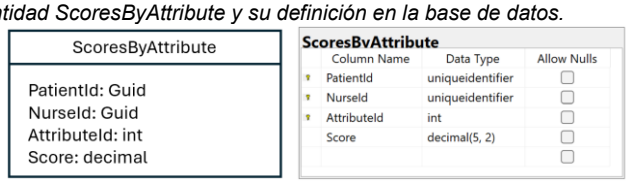
En esta fórmula solo se comparan los 2 usuarios que son más similares al usuario 1 y que además ya hayan calificado al ítem a; sin embargo, esta fórmula se puede extender utilizando tantos usuarios vecinos como se desee, pero se deberá tomar un máximo Top-N.

d) Almacenamiento de los scores

Después de calcular las predicciones para cada atributo, unirlas a las calificaciones reales de los usuarios y calcular los scores de las enfermeras, se procederá a almacenarlos en la BD ya que este procesamiento es sumamente pesado y necesita un tiempo considerable para ejecutarse. Al tener la información pre-calculada, se podrán realizar consultas al sistema de forma rápida y fácil. Se utilizará para ello, la entidad ScoresByAttribute y su equivalente de SQL (Coursera, 2025), en la Figura 8, se muestran estos componentes y sus propiedades.

Figura 8

Entidad ScoresByAttribute y su definición en la base de datos.



e) Weighted hybrid

La técnica hibrida de pesos ponderados o "Weighted hybrid" (Chiang, 2024) calcula la puntuación de predicción como el resultado de todos los enfoques de recomendación, considerándolos como variables en una combinación lineal, esta técnica asigna a cada uno de ellos un peso y suma los resultados ponderados. En (3) se muestra la fórmula para su cálculo, donde $r_{U,I}$ es el score final del ítem para el usuario, c es el número total de algoritmos, W_k es el peso asignado al algoritmo, recordando que la suma de los pesos debe ser 1, es decir, se deberá de convertir en porcentajes el valor de cada peso respecto al total y finalmente $r^k_{U,I}$ es la predicción del algoritmo k para el par de usuario e ítem.

Esta técnica permite modificar independientemente o agregar más técnicas de recomendación, permitiendo así el desacoplamiento e incremento iterativo de la visión de arquitectura propuesta.

$$r_{U,I} = \sum_{k=1}^{c} W_k \; r^k_{U,I} \tag{3}$$

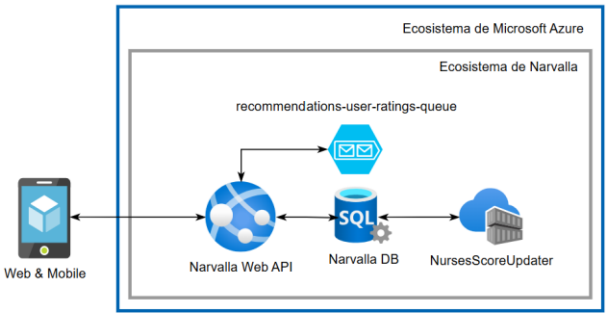
Dado que se busca la fácil integración con las demás áreas del sistema, se creará una vista (Coursera, 2025) en SQL llamada "Scores", que calculará el score general en base a las calificaciones de la tabla de ScoresByAttribute y los pesos de la tabla RatingAttributes para ser utilizados por la fórmula antes mencionada, esta vista contendrá además el id del usuario y de la enfermera.

f) Ejecución del proceso

Como ya se mencionó en la sección anterior, el procesamiento de recomendaciones puede llevar un tiempo considerable en ejecutarse; por tal motivo, se deberá de ejecutar como una tarea agendada en un horario poco concurrido del sistema para que no disminuya el desempeño de este. La hora podrá ser configurable, pero se propone que se ejecute a las 2 de la mañana. El proceso correrá en una instancia de Azure Container Apps (Microsoft, s. f.-a), servicio de Azure que permite el procesamiento en grandes volúmenes de información de forma económica (comparado a otros servicios), son configurables para agendar su ejecución y se integran fácilmente a los demás componentes del ecosistema de Narvalla, a este componente se le llamará "NursesScoreUpdater" y será programado en C# .NET usando Dapper (Learn Dapper, 2024) como librería de acceso a datos, ya que esta es ligera y eficiente a la hora de consultar información. Véase la Figura 9 de la arquitectura actualizada con este nuevo componente.

Año: 2024. Volumen: 1. Numero:11 ISSN: 2448-623X

Figura 9
Infraestructura incluyendo al NursesScoreUpdater.



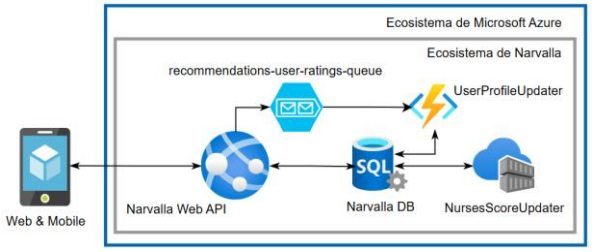
5) ARQ05: Afinación y optimización del modelo

A continuación, se presentarán los procesos de afinación del modelo de recomendación; gracias a ellos, se podrán ofrecer recomendaciones cada vez más precisas.

a) Procesamiento del domain event

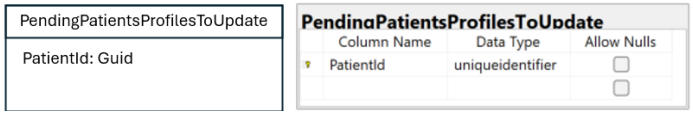
El proceso iniciará cuando un mensaje de ItemRatedByUserDomainEvent sea publicado en la cola recommendations-user-ratings-queue, esto ejecutará automáticamente un Azure Function (Microsoft, s. f.-a) de tipo QueueFunction (Lock, 2024) llamado "UserProfileUpdater" (véase la Figura 10).

Figura 10
Infraestructura incluyendo al UserProfileUpdater.



Este proceso almacenará mediante la nueva entidad "PendingPatientsProfilesToUpdate" (véase la Figura 11) al usuario analizado, esta tabla servirá como entrada para el componente NursesScoreUpdater, ya que solo se necesitan recalcular las similitudes y scores que involucren a los usuarios que han introducido nueva información de calificaciones al sistema, y después de cada ejecución, esta tabla será limpiada para el siguiente ciclo del proceso.

Figura 11
Entidad y tabla PendingPatientsProfilesToUpdate.



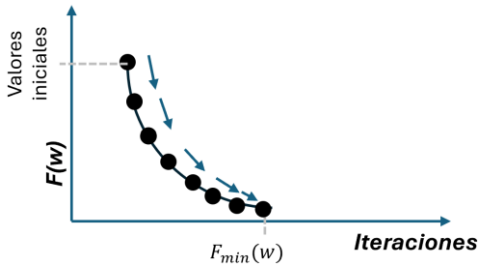
b) Afinación de pesos del weighted hybrid

Inicialmente, cada peso de los algoritmos utilizados para calcular el score tendrá el mismo valor, en este caso del 20%, para que los 5 sumen el 100%. Sin embargo, se necesitará un ajuste gradual de los mismos, ya que la mayoría de los usuarios, no consideran igual de importante la puntualidad que el conocimiento de la enfermera o su trato al paciente. Es decir, se buscará afinar el valor de cada peso, hasta que la diferencia de la calificación predicha y la real sea lo más cercana a cero.

Para poder obtener los pesos óptimos, es necesario intentar con diversas soluciones, hasta encontrar la mas adecuada, sin embargo, probar cada combinación de pesos hasta encontrar la mejor solución podría llegar a tomar días (método de fuerza bruta).

Para evitar este problema, es necesario el uso de técnicas de Machine Learning que permitan la reducción del margen de error, entre los pesos actuales y los pesos ideales de forma ordenada, utilizando métodos heurísticos para aproximarse a los pesos ideales. En la Figura 12, se muestra a w, la cual representa los valores de los pesos; $F(w)$ es la función de costo que evaluará los valores de w para obtener el margen de error; $F_{min}(w)$ la cual indica el valor óptimo cercano a cero para el margen de error; y el eje de las abscisas representa las iteraciones del proceso de optimización.

Figura 12
Representación gráfica de búsquedas heurísticas.

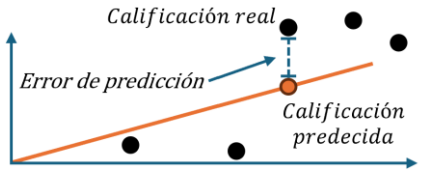


La función de costo es entonces la responsable de calcular la diferencia entre la predicción y la realidad de las calificaciones; una de las técnicas para llevar a cabo este proceso es la de "Root Mean Squared Error" (RMSE), esta función calcula la distancia promedio entre los valores predichos por un modelo y los valores reales en el conjunto de datos, esta distancia se conoce como el residuo o error de predicción (Sagi, 2014). Se describe en (4) la fórmula para calcular el RMSE, donde i representa a cada uno de esos pares de usuario-ítem que cuentan con una calificación real y una predicción dentro del sistema, mientras que N representa al número total de pares de usuario-ítem.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Calificacion\ Real_i - Prediccion\ Final_i)^2} \quad (4)$$

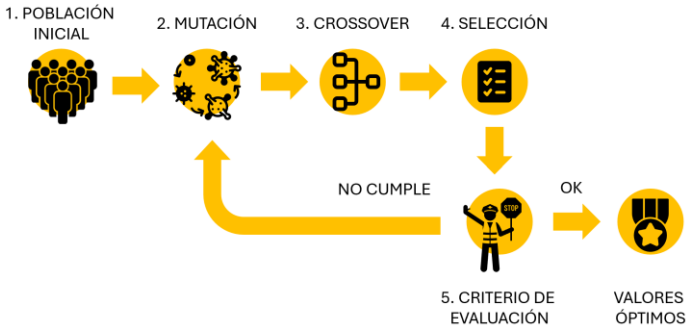
En la figura 13, se muestra de manera grafica el principio de esta técnica, donde el objetivo es obtener el error de predicción para su uso en la búsqueda heurística.

Figura 13
Representación gráfica de RMSE.



McCaffrey (2021) propone el uso de la técnica "Differential Evolution Optimization" (DEO); la cual tiene como función el imitar la evolución biológica, generando organismos y mutándolos para su mejor adaptabilidad al ecosistema; en este caso, el organismo a mutar es el vector compuesto por pesos de los atributos, el cual será mutado en varias iteraciones (McCaffrey, 2021). La figura 14 muestra de forma general, los pasos que los DEO deben de cumplir.

Figura 14
Etapas de un algoritmo Differential Evolution Optimization.



El primer paso es el de generar una población inicial (P) de N individuos con valores al azar, para nuestro caso, serán vectores de 5 elementos con valores del rango 0% a 100%, recordando que por ser vectores normalizados, la sumatoria de los 5 elementos deberá de ser de 100%, por ejemplo $V_i = [0.14, 0.24, 0.21, 0.13, 0.28]$.

Los pasos 2 y 3 consisten en mutación y crossover, para los cuales, por cada individuo V_i de P, se tomarán 3 más al azar V_a, V_b y V_c distintos a V_i, en (5) se muestra la fórmula para mutar vectores propuesta por McCaffrey (2021), y en la Tabla 4 se muestra una simulación de esta operación, donde las columnas W representan a los elementos del vector, es decir, los pesos (weights), y F es el factor de mutación (McCaffrey, 2021) con un valor de 80%.

$$V_m = V_a + F * (V_b - V_c) \tag{5}$$

Tabla 4
Ejemplo de cálculo de vector mutante

	W_1	W_2	W_3	W_3	W_5	*Suma*
V_a	0.07	0.14	0.18	0.22	0.39	1.00
V_b	0.10	0.26	0.31	0.12	0.21	1.00
V_c	0.06	0.02	0.32	0.33	0.27	1.00
V_m	*0.10*	*0.33*	*0.17*	*0.05*	*0.34*	*1.00*

Con el vector V_m calculado es ahora posible obtener un nuevo vector candidato $V_{candidato}$ mediante el crossover de V_i y V_m, este paso consiste en seleccionar algunos valores de ambos vectores mediante una variable *RANDOM* con valor al azar de entre 0 y 100%, si el valor es menor al umbral de crossover rate (CR) de 90%, se tomará el elemento de V_m, si el valor es mayor al CR, se tomará el valor de V_i. En la Tabla 5, se muestra esta operación, donde la mayoría de los elementos fueron tomados de V_m a excepción de W_5 que fue tomado de V_i.

Tabla 5
Ejemplo de cálculo del vector candidato.

	W_1	W_2	W_3	W_3	W_5
V_i	0.14	0.24	0.21	0.13	0.28
V_m	0.10	0.33	0.17	0.05	0.34
Random	**23%**	**42%**	**88%**	**39%**	**91%**
$V_{candidato}$	*0.10*	*0.33*	*0.17*	*0.05*	*0.28*

Notas: El $V_{candidato}$ también deberá de ser normalizado después de obtenerlo: $V_{candidato}$ = [0.11, 0.35, 0.18, 0.06, 0.30]. Si alguno de los elementos del $V_{candidato}$ es negativo, se deberá repetir el proceso de generación para V_i.

En el paso 4, correspondiente a la fase de selección, donde después de obtener el $V_{candidato}$, se calculará el RMSE de V_i y de V_c, si el RMSE de $V_{candidato}$ es menor al RMSE de V_i, se seleccionará y reemplazará V_i con $V_{candidato}$ en la lista de la población, de esta manera la población se irá mejorando en cada iteración, conteniendo resultados de menor RMSE.

Finalmente, después de recalcular y seleccionar la lista completa de la población, se verificará si entre la lista se encuentra un individuo que cumpla con el criterio de evaluación para ser considerado una solución óptima; Para este caso en particular, se buscará un RMSE de 0.1, es decir, $F_{min}(w) = 0.1$.

Si este valor no se encuentra entre las soluciones de la población, entonces se volverá a ejecutar el proceso desde el paso 2. En caso de que después de 100 iteraciones esta condición no se haya cumplido, se considerará como solución óptima aquella con el menor RMSE, y se esperará a la siguiente ejecución del programa para continuar con su mejora.

c) Integración con el Sistema recomendador

Para ejecutar este proceso, es necesario que ciertos parámetros sean configurables en el sistema. La variable $F_{min}(w)$ se denominará RmseThreshold y tendrá un valor por defecto de 0.1. El tamaño de la población inicial se configurará como InitialPopulationSize, con un valor predeterminado de 50. Asimismo, el factor de mutación F se llamará MutationFactor, con un valor por defecto de 0.8, y la tasa de cruce será conocida como CrossoverRate, configurada con un valor de 0.9 por defecto. El número máximo de iteraciones se definirá como MaxIterations y su valor predeterminado será de 100. Para todas las variables de configuración, se deberán realizar simulaciones agendadas que permitan analizar el comportamiento del sistema con distintos valores.

De manera inicial, se asignarán pesos iguales a cada atributo en la tabla RatingAttributes, y se generarán los scores por atributo utilizando estos pesos como entrada, almacenando los resultados en la tabla ScoresByAttribute (véase la sección ARQ04).

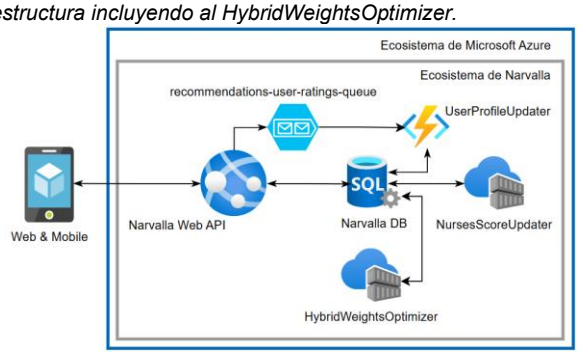
Una vez cumplidos los prerrequisitos, se procederá a preparar los datos necesarios para el procesamiento. Para ello, se cargarán en memoria las calificaciones reales almacenadas en la tabla Rating, agrupadas por promedio en caso de que exista más de un registro para cada par usuario-ítem. Después, se cargarán en memoria las predicciones de la tabla ScoresByAttribute, solo se cargarán aquellas predicciones que hagan match con los pares de usuario-ítem existentes del resultado del paso anterior.

Después de obtener una solución óptima para los nuevos pesos, estos se actualizarán en la tabla RatingAttributes. Este

Año: 2024. Volumen: 1. Numero:11 ISSN: 2448-623X

proceso se realizará cada fin de mes, debido al poder computacional necesario para llevarlo a cabo, además, después de la primera iteración no será tan efectiva, ya que los pesos no variarán demasiado. Al igual que para el proceso del NursesScoreUpdater, esta operación se llevará a cabo en un Azure Container Apps y su nombre será HybridWeightsOptimizer; se presenta en la Figura 15 la infraestructura actualizada.

Figura 15
Infraestructura incluyendo al HybridWeightsOptimizer.

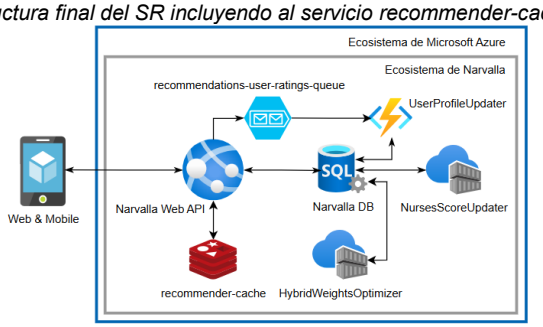


Con estos pasos se asegura la mejora continua del proceso de recomendación, nuevamente considerando la escalabilidad y extensión iterativa del sistema.

6) ARQ06: Integración de los scores al sistema

La plataforma de Narvalla utilizará la vista Scores como punto de acceso a los datos procesados por el SR. Uno de los beneficios de usar esta vista es que el cálculo será dinámico, siempre calculando el score final de cada par de usuario-ítem con los valores de los pesos más actuales. Dado que la información se recalcula cada 24 horas, se utilizará un sistema de cache distribuido (Educative & Educative, s. f.) como lo es Azure Redis Cache (Microsoft, s. f.-a). Dentro de las aplicaciones que consuman esta vista, las entradas en el cache, referentes al SR deberán de tener una vida de 6 horas. En la Figura 16 se muestra el diseño final de la infraestructura que contiene este nuevo componente llamado recommender-cache.

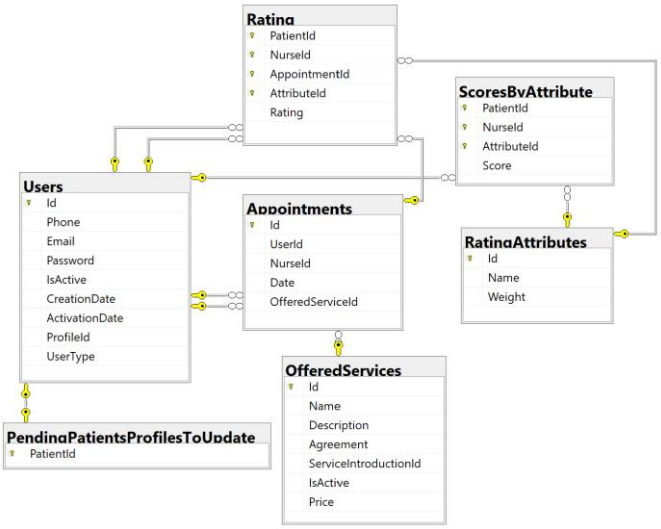
Figura 16
Infraestructura final del SR incluyendo al servicio recommender-cache.



7) Búsqueda con recomendaciones

A continuación, se presenta el diagrama SQL de la Figura 17 con las tablas más relevantes para el SR y sus relaciones. Nótese que las entidades Paciente y Enfermera se almacenan en la misma tabla de Users, pero el UserType es diferente. Gracias a ellas y a la vista Scores, se podrán generar múltiples consultas a la base de datos, como, por ejemplo, obtener un listado de las 100 mejores enfermeras para recomendarle al paciente x que busca un servicio y; el resultado deberá ser ordenado de mayor a menor según su clasificación. Más tablas del sistema pueden unirse para poder realizar filtros más específicos en el resultado, como lo son, ubicación, disponibilidad, entre otras.

Figura 17
Diagrama de tablas y relaciones del SR.



Con esta pieza final se logra diseñar una solución para todos los requisitos solicitados por Narvalla.

E. Evaluación de riesgos

Una vez obtenido el análisis de requerimientos y diseño de arquitectura, es posible detectar riesgos para el proyecto, cabe mencionar que para muchos ya se ha definido una acción de mitigación. A continuación, se presenta en la Tabla 6, una matriz de riesgos como la describe Coordinacae (2024).

Tabla 6
Matriz de riesgos.

Riesgo	*Acción de mitigación*
Inicio en frío	Solicitar explícitamente por retroalimentación a usuarios nuevos y actuales. Priorizar el desarrollo del proceso de evaluación del servicio para recopilar datos aun y cuando el SR no este completado.
Procesamiento pesado	Se ejecutará asíncronamente y en horario de bajo impacto para no afectar la operación. Al usar computación en la nube, se podrá escalar el procesamiento del servicio fácilmente de ser necesario.
Latencia por recalcular scores	Se utilizará una tabla con valores de scores pre-calculados para su fácil obtención. Además, se utilizará cache local y distribuido en las web api de Narvalla.
Costos exponenciales en Azure	Al ser una plataforma escalable, también lo pueden ser sus costos, se deberán de configurar límites de escalabilidad, alertas de consumo y límites de presupuesto en la plataforma.
Escalabilidad del filtrado user-based	Para que no crezca exponencialmente el cálculo de similitudes entre usuarios, se estableció un Top-N de vecinos similares para solo calcular los más cercanos, además del uso de la tabla PendingPatientsProfilesToUpdate que limitará cada ejecución del proceso.

Cuellos de botella en SQL	Una vez definidas todas las consultas que se realizarán para esta implementación, se detectarán campos que necesiten ser indexados. Y de ser necesario, se considerará el uso de tablas en memoria de SQL.
Fallos en la programación	Se deberá de utilizar pruebas unitarias (Schmitt, 2021) y pruebas de integración (Schmitt, 2021) en cada uno de los componentes, para reducir al mínimo los errores de programación.

III. RESULTADOS Y DISCUSIÓN

A. Resultados obtenidos

Vpadmin (2023) menciona varios artefactos que pueden ser generados como resultado de la etapa de arquitectura al utilizar TOGAF o ADM, metodologías de arquitectura de software. El primero y más importante obtenido en este documento es la declaración de una visión de arquitectura sólida, que permita describir el estado futuro de la organización después de implementar el proyecto. El segundo artefacto obtenido es la arquitectura de los datos, el cual se presenta en la Figura 17, mientras que el artefacto de arquitectura de aplicaciones se muestra en la Figura 16, donde cada componente es indicado, así como sus interacciones principales. Después, el artefacto de evaluación de riesgos, mostrado en la Tabla 6. Por último, el artefacto de especificación de requisitos de arquitectura fue abordado gracias a los elementos REQ01, REQ02, REQ03, REQ04, REQ05 y REQ06, mismos que permitieron la creación del artefacto de arquitectura tecnológica, en donde gracias a los ítems ARQ01, ARQ02, ARQ03, ARQ04, ARQ05 y ARQ06, fue posible definir las técnicas, algoritmos, procesos y herramientas para la futura ejecución del proyecto.

B. Resultados teóricos esperados

Požar (s. f.) indica que las empresas que utilizan un SR incrementan sus ventas desde un 35% hasta un 70%, que aun con una expectativa pesimista, un 20% de incremento en ventas se considerará un éxito para el proyecto. Además de permitir obtener una mejor comprensión de las necesidades de los clientes y con ello planear estrategias a largo plazo. Finalmente, será posible clasificar y perfilar a los pacientes para realizar campañas de marketing focalizadas, disminuyendo así el costo de fidelización de clientes (Požar , s. f.).

Al utilizar un enfoque de computación distribuida en la nube es posible tomar ventaja de los beneficios que ofrece la plataforma de Microsoft Azure desde el día 1, los cuales son: escalabilidad, seguridad, ahorro de costos de implementaciones, disponibilidad, flexibilidad, entre otros (Google, s. f.).

C. Discusión de la propuesta

Las ventajas y beneficios obtenidos al ejecutar la visión de la arquitectura propuesta ya han sido mencionadas, desacoplamiento, escalabilidad, evolución iterativa, entre otras, pero ¿cuáles son las desventajas de este enfoque? Una de ellas es la complejidad inicial del diseño; cada iteración deberá ser planeada con cautela para no generar dependencias innecesarias. Esto incrementará el tiempo de cada iteración significativamente. Además, al haber múltiples componentes independientes en juego, la trazabilidad, monitoreo y mantenimiento del sistema escalará en complejidad y costes.

Si bien la computación en la nube facilita la integración entre componentes desacoplados, también genera una completa dependencia hacia los proveedores de servicios, como lo es en este caso Microsoft.

Además, se debe considerar que la curva de aprendizaje para los nuevos integrantes del equipo técnico resultará mucho mayor, ya que tendrán que comprender dónde se encuentran y cómo se comunican los componentes desacoplados del sistema.

Los beneficios obtenidos superan por mucho los problemas que esta implementación conlleva, pero no por ello dejan de ser riesgos que deberán ser mitigados. Mediante planes de contingencia y una correcta documentación de los procesos y componentes de la plataforma se logrará disminuir estos riesgos de forma significativa.

Una alternativa a la arquitectura propuesta es utilizar un enfoque de sistemas monolíticos, los cuales solucionan muchos de los problemas planteados; sin embargo, al resolver estos problemas, generan nuevos, como lo son la dificultad para integrar nuevas funcionalidades, ya que todos los procesos residirán en un mismo lugar, lo cual incrementará la complejidad del código, su mantenimiento y escalabilidad.

Finalmente, Microsoft Azure y otras plataformas SaaS (Google, s. f.) ofrecen servicios de SR empaquetados y listos para utilizarse a un bajo costo en comparación al costo que conlleva la investigación e implementación por cuenta propia. Sin embargo, este tipo de servicios ofrece muy poca personalización, lo cual no permite adecuar una solución a la medida como la deseada para Narvalla.

D. Relevancia para proyectos futuros

Mucho se ha mencionado la fácil integración de nuevas funcionalidades al sistema mediante esta visión de arquitectura, por lo que una forma sencilla de abordar el tema es mediante un ejemplo. Supongamos que llega un nuevo requerimiento para el SR, en el cual, cuando un paciente busque agendar el servicio de una enfermera a la que contrata recurrentemente, pero esta se encuentra de vacaciones, el usuario podría visualizar en el perfil de la enfermera, un listado de otras cuidadoras muy similares a ella, es decir que compartan atributos y calificaciones parecidos, para ello, se necesitará generar una implementación de filtros colaborativos item-to-item, esto implicaría generar un proceso similar al del SR user-based, pero ahora sería item-based, por lo que se necesitaría calcular la similitud entre los ítems, recalcular sus similitudes cada vez que nuevas calificaciones entren al sistema, etc.

Otra posible línea de evolución del sistema consiste en ampliar los métodos de recomendación integrados en el cálculo del score basado en pesos. Actualmente, el modelo considera cinco características específicas, pero es plenamente escalable tanto en la incorporación de nuevas propiedades como en la adición de enfoques complementarios. Por ejemplo, podrían incluirse puntuaciones basadas en la popularidad de la enfermera, su nivel académico, certificaciones obtenidas, o la experiencia específica relacionada con el tipo de servicio solicitado por el paciente, entre otros factores relevantes, cuyo cálculo no esté relacionado con calificaciones de usuarios.

IV. CONCLUSIONES

Se concluye que la aplicación de técnicas de inteligencia artificial orientadas a recomendaciones personalizadas resulta viable para la plataforma de Narvalla.

Se logró determinar que, mediante una visión de arquitectura orientada al desacoplamiento de componentes, eventos de dominio, evolución iterativa y procesamiento distribuido en la nube, es posible escalar de forma fácil y segura las funcionalidades presentadas y futuras para el sistema.

Se comprueba que cada uno de los sub-objetivos presentados es asequible desde un punto de vista técnico y funcional.

V. AGRADECIMIENTOS

Los autores agradecen al equipo de trabajo de Narvalla por su participación en las sesiones de recolección de requerimientos, así como a familiares, amigos y colegas que les apoyaron en la revisión de este documento.

VI. REFERENCIAS

Agile Alliance. (2024, 10 diciembre). *What is Agile? | Agile 101 | Agile Alliance*. Agile Alliance |. https://www.agilealliance.org/agile101/

Amazon. (s. f.). *Front End vs Back End - Difference Between Application Development - AWS*. Amazon Web Services, Inc. Recuperado 11 de abril de 2025, de https://aws.amazon.com/compare/the-difference-between-frontend-and-backend/

Angular. (s. f.). *Angular*. Recuperado 11 de abril de 2025, de https://v17.angular.io/guide/what-is-angular

Beer, S. (2024, 25 octubre). *Custom Development Software vs. Out-of-the-Box Software*. Clarity Ventures. Recuperado 11 de abril de 2025, de https://www.clarity-ventures.com/articles/custom-development-vs-out-of-the-box-software

Bootstrap. (s. f.). *Bootstrap*. Recuperado 11 de abril de 2025, de https://getbootstrap.com/

ByteHide. (2024, 8 octubre). *Essential .NET libraries every developer should know*. DEV Community. https://dev.to/bytehide/essential-net-libraries-every-developer-should-know-1lp3

Caldera Sánchez, C. J., Uribe Agundis, D., Cuan Durón, E., & Urquizo Barraza, E. (2015). Sistema recomendador con implementación “Software as a Service”. *Memorias del Congreso Internacional de Investigación Academia Journals Celaya 2015*, *7*(4), ISSN 1946-5351. https://academia-journals.squarespace.com/s/Celaya-Memorias-ONLINE-2015-Tomo-04.pdf

Chiang, J. (2024, 14 marzo). 7 Types of Hybrid Recommendation System - Analytics Vidhya - Medium. Medium. https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8

Coordinacae. (2024, 16 abril). Matriz de riesgos | Optimiza tu PRL con una matriz eficaz. Coordinaplus. Recuperado 14 de abril de 2025, de https://www.coordinacae.com/blog/matriz-de-riesgos/

Coralogix. (2023, 28 febrero). *What are recommender systems? use cases, types & techniques*. https://coralogix.com/ai-blog/what-are-recommender-systems-use-cases-types-and-techniques/

Coursera. (2025, 9 abril). *SQL Glossary: Your Ultimate Guide to SQL Terms*. Coursera. https://www.coursera.org/resources/sql-terms

Culibrk, A. (2024, 19 enero). How to solve the cold start problem in recommender systems - Things Solver. *Things Solver*. https://thingsolver.com/blog/the-cold-start-problem/

Educative, & Educative. (s. f.). *Grokking the Modern System Design Interview*. Educative. https://www.educative.io/courses/grokking-the-system-design-interview/system-design-the-distributed-cache

García, Á. (2021, 10 junio). Arquitectura Evolutiva para Desarrolladores - Apiumhub. *Apiumhub*. Recuperado 11 de abril de 2025, de https://apiumhub.com/es/tech-blog-barcelona/arquitectura-evolutiva-para-desarrolladores/

Google. (s. f.). *Advantages of Cloud Computing | Google Cloud*. Google Cloud. Recuperado 12 de abril de 2025, de https://cloud.google.com/learn/advantages-of-cloud-computing

Huet, P. (2022, 24 agosto). Arquitectura de software: Qué es y qué tipos existen. *OpenWebinars.net*. https://openwebinars.net/blog/arquitectura-de-software-que-es-y-que-tipos-existen/

IBM. (2025, 13 marzo). Aprendizaje automático. *IBM*. Recuperado 11 de abril de 2025, de https://www.ibm.com/mx-es/think/topics/machine-learning

Kurama, V. (2025, 22 enero). *What is collaborative filtering: A simple introduction*. Built In. https://builtin.com/data-science/collaborative-filtering-recommender-system

Learn Dapper. (2024, 17 octubre). *Welcome to Learn Dapper*. ZZZ Projects. Recuperado 12 de abril de 2025, de https://www.learndapper.com/

Lock, A. (2024, 27 agosto). *Using Azure Storage Queue messages with Azure Functions and [QueueTrigger]*. Andrew Lock | .NET Escapades. https://andrewlock.net/using-azure-storage-queues-with-azure-functions-and-queuetrigger/

Lucidchart. (s. f.). *Qué es un diagrama entidad-relación*. Recuperado 11 de abril de 2025, de https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Martin, R. C. (2012, 13 agosto). *Clean Coder Blog*. Recuperado 11 de abril de 2025, de https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

McCaffrey, J. (2021, 7 septiembre). *Differential Evolution Optimization -- Visual Studio Magazine*. Visual Studio Magazine. Recuperado 12 de abril de 2025, de https://visualstudiomagazine.com/Articles/2021/09/07/differential-evolution-optimization.aspx

Microsoft. (s. f.-a). *Directory of Azure Cloud Services | Microsoft Azure*. Recuperado 11 de abril de 2025, de https://azure.microsoft.com/en-us/products

Microsoft. (s. f.-b). *Get to Know Azure | Microsoft Azure*. Recuperado 11 de abril de 2025, de https://azure.microsoft.com/en-us/explore/

NASA. (s. f.). *What is artificial intelligence? - NASA*. Recuperado 11 de abril de 2025, de https://www.nasa.gov/what-is-artificial-intelligence/

Neto, J. L. (2023, 7 marzo). How we used Event Storming Meetings for enabling Software Domain-Driven Design. *Medium*. https://medium.com/building-inventa/how-we-used-event-storming-meetings-for-enabling-software-domain-driven-design-401e5d708eb

Oracle. (2020, 24 noviembre). *What Is a Database?* https://www.oracle.com/database/what-is-database/

Pedamkar, P. (2023, 12 junio). *ASP.NET vs C#*. EDUCBA. Recuperado 11 de abril de 2025, de https://www.educba.com/asp-net-vs-c-sharp/

Postman. (s. f.). *API Glossary: API & Programming Terminology | Postman*. Recuperado 11 de abril de 2025, de https://www.postman.com/api-glossary/

Požar, N. (s. f.). *Recommendation systems in E-commerce: What’s the thing you’ve never known, but always wanted to?* Recuperado 12 de abril de 2025, de https://www.be-terna.com/insights/recommendation-systems-in-e-commerce-whats-the-thing-youve-never-known-but-always-wanted-to

Sagi, O. S. (2014, septiembre). *Model performance metrics for regression models | Pecan Help Center*. Pecan. Recuperado 22 de junio de 2025, de https://help.pecan.ai/en/articles/6456388-model-performance-metrics-for-regression-models

Schmitt, J. (2021, 3 diciembre). *Unit testing vs integration testing*. CircleCI. https://circleci.com/blog/unit-testing-vs-integration-testing/

Tapereal. (2024, 21 agosto). 6 Strategies to Solve Cold Start Problem in Recommender Systems. *Tapereal*. Recuperado 14 de abril de 2025, de https://web.tapereal.com/blog/6-strategies-to-solve-cold-start-problem-in-recommender-systems/

Vpadmin. (2023, 10 octubre). *What is Architecture Artifacts in TOGAF ADM - Visual Paradigm TOGAF*. Visual Paradigm TOGAF. https://togaf-visual--paradigm-com.translate.goog/2023/10/10/what-is-architecture-artifacts-in-togaf-adm/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

Winand, M. (s. f.). *Sentencias Top-N: traer solamente los primeros N registros*. Use The Index, Luke! Recuperado 11 de abril de 2025, de https://use-the-index-luke.com/es/sql/resultados-parciales/sentencia-top-n