

# Desarrollo de una aplicación web para la selección de pedidos con sistema de recomendación para Beer & Coffee Company

Ramsés de Jesús Hidalgo Guevara

Universidad del Valle  
Escuela de Ingeniería en Sistemas y Computación  
Ingeniería en Sistemas y Computación  
Tuluá - Valle del Cauca  
Febrero - Junio 2024

# Tabla de Contenido

<b>1. Introducción</b>	<b>8</b>
1.1. Formulación del Problema . . . . .	8
1.2. Descripción del Problema . . . . .	8
1.3. Objetivos . . . . .	9
1.3.1. Objetivo General . . . . .	9
1.3.2. Objetivos Específicos y Resultados Esperados . . . . .	9
1.4. Metodología . . . . .	9
1.4.1. Recolección de Requisitos (2 semanas) . . . . .	10
1.4.2. Diseño del Sistema (4 semanas) . . . . .	10
1.4.3. Desarrollo e Implementación (12 semanas) . . . . .	10
1.4.4. Despliegue (1 semana) . . . . .	10
1.4.5. Pruebas y Validación (2 semanas) . . . . .	10
1.5. Información sobre los capítulos . . . . .	11
<b>2. Justificación y alcance del proyecto</b>	<b>12</b>
2.1. Justificación . . . . .	12
2.2. Alcance del Proyecto . . . . .	12
2.2.1. Características Principales . . . . .	13
2.2.2. Exclusiones del Proyecto . . . . .	13
2.2.3. Limitaciones y Suposiciones . . . . .	13
<b>3. Marco Referencial</b>	<b>15</b>
3.1. Marco Teórico . . . . .	15
3.1.1. Sistemas de Recomendación . . . . .	15
3.1.2. Sistema de Recomendación Basado en Contenido (CBF) . . . . .	16
3.1.3. Sistemas de Recomendación Híbridos . . . . .	16
3.1.4. Técnicas de Recomendación Basada en Grafos . . . . .	16
3.1.5. Métricas de sistemas de recomendación basado en filtrado colaborativo (CF) . . . . .	16
3.1.6. Aplicaciones Web . . . . .	17
3.1.7. Servicios Web . . . . .	17
3.1.8. Arquitectura de Software . . . . .	17
3.1.9. Psicología del Consumidor . . . . .	18
3.2. Marco Conceptual . . . . .	18
3.2.1. Sistemas de recomendación . . . . .	19
3.2.2. Arquitectura de microservicios . . . . .	19
3.2.3. Paradoja de Elección . . . . .	19
3.2.4. Experiencia del Usuario (UX) . . . . .	19

3.2.5. Recomendaciones basdas en grafos . . . . .	19
3.2.6. Algoritmos de filtrado colaborativo basada en usuarios . . . . .	19
<b>4. Diseño e implementación</b>	<b>20</b>
4.1. Diseño e Implementación de la Interfaz . . . . .	20
4.1.1. Prototipo e interfaces de usuario . . . . .	20
4.2. Arquitectura de la Aplicación . . . . .	26
4.2.1. Introducción . . . . .	26
4.2.2. Proceso de Funcionamiento de la Aplicación . . . . .	27
4.3. Diseño e implementación del sistema de recomendación . . . . .	29
4.3.1. Elección del Sistema de Recomendación de Filtrado Colaborativo . . . . .	29
4.3.2. Recomendaciones en Tiempo Real . . . . .	30
4.3.3. Funcionamiento de los grafos en el sistema de recomendación . . . . .	30
4.3.4. Modelo de base de datos NEO4J . . . . .	31
4.3.5. Implementacion de algoritmos . . . . .	33
4.4. Servicio de Búsqueda . . . . .	43
4.4.1. Implementación . . . . .	43
4.5. Servicio de órdenes . . . . .	46
4.5.1. Implementación . . . . .	46
<b>5. Despliegue</b>	<b>48</b>
5.1. Proceso de Despliegue de la Aplicación Web . . . . .	48
5.1.1. Configuración de Docker . . . . .	48
5.1.2. Configuración de Nginx . . . . .	48
5.1.3. Integración con Ngrok . . . . .	48
5.1.4. Flujo de Despliegue . . . . .	49
<b>6. Pruebas</b>	<b>50</b>
6.1. Pruebas de Usabilidad y Aceptación . . . . .	50
6.1.1. Introducción . . . . .	50
6.1.2. Metodología . . . . .	50
6.1.3. Pruebas de Usabilidad . . . . .	50
6.1.4. Pruebas de Aceptación . . . . .	51
6.1.5. Análisis de los Resultados Más Bajos . . . . .	53
<b>7. Conclusiones</b>	<b>54</b>
<b>8. Trabajos futuros</b>	<b>56</b>
8.0.1. Mejoramiento del Sistema de Recomendaciones . . . . .	56
8.0.2. Optimización del Sistema de Búsqueda . . . . .	56
8.0.3. Migración a la Nube . . . . .	56
8.0.4. Mejora de la Interfaz de Usuario . . . . .	57
8.0.5. Integración de Pagos en Línea . . . . .	57
8.0.6. Sistema de Registro y Login Más Completo . . . . .	57
8.0.7. Mejorar el Panel de Administradores . . . . .	57
8.0.8. Evaluación y Optimización Continuas . . . . .	57
<b>Referencias</b>	<b>58</b>



# Lista de Figuras

3.1. Arquitectura de microservicios y monolítica . . . . .	18
4.1. Prototipo e interfaz login de usuario para cliente y administrador . . . . .	21
4.2. Prototipo e interfaz inicio de la página catálogo de productos . . . . .	22
4.3. Prototipo e interfaz inicio de cliente . . . . .	23
4.4. Prototipo e interfaz para cliente de detalles del producto . . . . .	24
4.5. Prototipo e interfaz de lista de ordenes cliente . . . . .	25
4.6. Panel de administración y páginas . . . . .	26
4.7. Arquitectura general de la aplicación . . . . .	27
4.8. Diagrama de secuencias de la aplicación . . . . .	29
4.9. Diagrama de secuencias del diseño sistema de recomendaciones . . . . .	31
4.10. Modelo de la base de datos con nodos e interacciones guardadas en neo4j . . . . .	31
4.11. Interacciones almacenadas en la base de datos Neo4j . . . . .	35
4.12. Nodo usuario y producto guardados en Neo4j . . . . .	36
4.13. Fórmula para calcular la correlacion de Pearson.[35] . . . . .	37
4.14. Modelo de grafo del sistema indicando productos que se recomienda al usuario . . . . .	39
4.15. Recomendaciones de “Ramses III.” <sup>en</sup> Neo4j browser . . . . .	40
4.16. Recomendaciones de “Alejandro Magno.” <sup>en</sup> Neo4j browser . . . . .	40
4.17. Recomendaciones de “gengis khan.” <sup>en</sup> Neo4j browser . . . . .	41
4.18. Resultados de los productos más populares del grafo anterior en Neo4j Browser . . . . .	42
4.19. Diagrama de flujo del proceso de búsqueda en el servicio de búsqueda . . . . .	45
4.20. Captura de pantalla prueba Curl a la Api de búsqueda . . . . .	46
4.21. Diagrama de flujo del proceso del algoritmo de ordenes . . . . .	47
6.1. Gráfica de la precisión de los detalles del producto . . . . .	51
6.2. Gráfica de la utilidad de las recomendaciones . . . . .	51
6.3. Gráfica de la satisfacción con las recomendaciones . . . . .	52
6.4. Gráfica de la influencia de las recomendaciones . . . . .	52

# Resumen

Este proyecto se centra en el desarrollo de una aplicación web destinada a mejorar la experiencia de selección de pedidos en Beer & Coffee Company. El establecimiento, conocido por su amplia variedad de cervezas artesanales y bebidas de café, donde esta variedad y cantidad de opciones para elegir dificulta su toma de decisiones, reduciendo su satisfacción con el establecimiento. La aplicación busca simplificar el proceso de elección mediante la implementación de un sistema de recomendaciones basado en filtrado colaborativo. Este sistema sugiere productos a los usuarios basándose en sus interacciones previas y en las preferencias de otros clientes con gustos similares. Para lograr esto, se utiliza una base de datos de grafos (Neo4j), que permite registrar y analizar las interacciones entre usuarios y productos. El algoritmo de filtrado colaborativo empleado se adapta para identificar usuarios con patrones de comportamiento similares y recomendar productos que estos han disfrutado.

La arquitectura de la aplicación se basa en microservicios. Los servicios incluyen autenticación, gestión de productos, búsqueda, gestión de pedidos, interacciones y recomendaciones. Cada servicio opera de manera independiente y se comunica a través de APIs, facilitando la actualización y el escalado del sistema sin afectar otros componentes. El uso de Nginx como API Gateway centraliza la gestión del tráfico y optimiza el rendimiento general. El desarrollo del proyecto siguió una metodología ágil, con etapas definidas: recolección de requisitos, diseño del sistema, desarrollo e implementación, despliegue y pruebas.

En la fase de diseño del sistema, se crearon mockups para la interfaz de usuario, y se diseñaron la arquitectura del sistema y el modelo de datos. La fase de desarrollo e implementación involucró el desarrollo concurrente del frontend y backend. El despliegue de la aplicación se realizó en un entorno de prueba, utilizando Docker y Ngrok para facilitar el acceso y pruebas de los usuarios. La configuración de Docker permitió la contenedorización de los servicios, asegurando una implementación consistente y eficiente. Ngrok se utilizó para exponer localmente la aplicación y realizar pruebas remotas.

Finalmente, se llevaron a cabo pruebas de usabilidad y comportamiento con usuarios. Estas pruebas tuvieron como objetivo evaluar y validar el sistema, identificando áreas de mejora y asegurando que la aplicación cumpliera con las expectativas de los usuarios. Se realizaron encuestas, donde se recopiló retroalimentación directa sobre la experiencia del usuario, la funcionalidad del sistema y la efectividad de las recomendaciones.

## Keywords

Sistema de Recomendación, Filtrado Colaborativo, Microservicios, Aplicación Web, Interfaz de Usuario

# Capítulo 1

## Introducción

### 1.1. Formulación del Problema

¿Cómo desarrollar una aplicación web con sistema de recomendación para la selección de pedidos para los clientes de Beer & Coffee Company?

### 1.2. Descripción del Problema

El establecimiento Beer & Coffee Company, con más de siete años en la ciudad de Tuluá, se ha consolidado como un referente en la industria gastrobar local [1]. Su principal distintivo es la venta de cerveza artesanal y bebidas de café, destacándose por ofrecer más de 300 tipos de productos, incluyendo cervezas artesanales, bebidas frías y bebidas calientes. Además, su carta incluye más de 10 tipos de comidas y 18 cócteles, ofreciendo una gran variedad para todos los gustos [1].

Sin embargo, esta extensa variedad de opciones, que en principio parece ser una ventaja competitiva, se ha convertido en una fuente de incertidumbre e inseguridad para los clientes. Este fenómeno, conocido como la paradoja de la elección, sugiere que cuando los consumidores se enfrentan a demasiadas opciones, pueden sentirse abrumados y menos satisfechos con sus decisiones [2].

El problema radica en que los clientes, al enfrentarse a un menú tan amplio y variado, experimentan dificultades para tomar decisiones informadas y seguras debido a que la sobreabundancia y poco detalle de las opciones genera una sensación de inseguridad y duda sobre la calidad de los productos elegidos [3]. Esto puede resultar en una disminución de la satisfacción general del cliente con su experiencia en Beer & Coffee Company.

Para abordar esta problemática, se ha identificado la necesidad de desarrollar e implementar una aplicación web orientada al cliente. Este software debe permitir a los clientes ver un catálogo de productos con sus detalles de manera organizada y clara, con secciones de productos recomendados y populares, así como una función de búsqueda. De esta manera, se podrá simplificar la elección de los clientes y proporcionarles una experiencia más agradable y satisfactoria.

## 1.3. Objetivos

### 1.3.1. Objetivo General

Desarrollar una aplicación web para la selección de pedidos con sistema de recomendación para Beer & Coffee Company.

### 1.3.2. Objetivos Específicos y Resultados Esperados

A continuación, se presenta la relación entre los objetivos específicos y los resultados esperados:

Objetivo Específico	Resultado Esperado
Analizar y definir los requerimientos	Requerimientos funcionales y no funcionales bien analizados y definidos.
Diseñar la arquitectura del sistema con su modelo de datos	Una arquitectura diseñada adecuadamente. Diagramas del modelo de base de datos.
Diseñar y desarrollar una interfaz responsiva que facilite a los clientes la exploración de los productos	Una interfaz de usuario responsive fácil de usar para el usuario. Documentación del diseño de interfaz de usuario, prototipos de pantalla.
Diseñar e implementar el sistema de recomendaciones de filtrado colaborativo	Un sistema de recomendación que, a través de algoritmos de filtrado colaborativo, genere recomendaciones. Documentación técnica de los algoritmos de recomendación, e informes de implementación del sistema.
Realizar la construcción y completa integración de los servicios de búsqueda y órdenes en la aplicación web	Servicios de búsqueda y órdenes integrados y funcionales.
Realizar pruebas de aceptación y usabilidad de la aplicación web	Sistema probado y validado, con un informe de pruebas de aceptación y usabilidad completado.

Cuadro 1.1: Relación entre Objetivos Específicos y Resultados Esperados

## 1.4. Metodología

Para la elaboración del proyecto se adoptaron principios ágiles, lo que permitió ajustar los requisitos y objetivos a medida que avanzaba el desarrollo. Se aplicaron principios de adaptabilidad, flexibilidad y retroalimentación constante[4]. La elección de una arquitectura de servicios permitió modularidad y escalabilidad, centrándose en la construcción de un servicio independiente del resto. Los textos generales del sistema de recomendaciones se elaboraron sobre la base de diversas fuentes de información, que incluyen artículos científicos, libros y tutoriales, los cuales guiaron y apoyaron el desarrollo[5][6][7][8]. Como resultado del análisis de la información recopilada, se implementaron algoritmos de recomendación de filtrado colaborativo basado en usuarios y algoritmos que muestran los productos populares, utilizando la potencia de las bases de datos centradas en grafos, como Neo4j. Estas bases de datos se alimentan con las interacciones de los usuarios en tiempo real. También se utilizaron algoritmos de procesamiento de lenguaje natural con la librería Spacy en el servicio de búsqueda y se emplearon WebSockets para la realización de pedidos. El proyecto se diseñó, implementó y desplegó de forma autónoma. Se mantuvo retroalimentación con el director del proyecto semanalmente y se presentó el progreso cada dos semanas. El cliente no tuvo acceso al proyecto hasta la etapa de pruebas.



**1.4.1. Recolección de Requisitos (2 semanas)**

- Definir requisitos funcionales y no funcionales.
- Identificar componentes con casos de uso e historias de usuario para trazar la funcionalidad.

**1.4.2. Diseño del Sistema (4 semanas)**

- Diseñar wireframes para la interfaz de usuario.
- Diseñar la arquitectura de servicios del sistema en general que permita recomendaciones, búsquedas y órdenes, así como servicios de acceso a usuarios y gestión de productos.

**1.4.3. Desarrollo e Implementación (12 semanas)**

- Desarrollar el frontend de manera concurrente con el backend, utilizando tecnologías como React.js, Spring Boot, Flask y Node.js.

**1.4.4. Despliegue (1 semana)**

- Desplegar la aplicación web en un entorno de prueba, usando Docker y Ngrok para facilitar el acceso de los usuarios.

**1.4.5. Pruebas y Validación (2 semanas)**

- Realizar pruebas de usabilidad y comportamiento con 30 usuarios mediante Google Forms.
- Evaluar los resultados de las pruebas, dejándolos pendientes para una etapa futura.

## 1.5. Información sobre los capítulos

En la siguiente tabla se muestra una corta descripción de cada capítulo del documento y se explica qué objetivos específicos aborda.

Capítulo	Descripción	Objetivo específico
Capítulo 2: Justificación y alcance del proyecto	Expone las razones del proyecto, estableciendo las funcionalidades y las restricciones.	Ninguno
Capítulo 3: Marco referencial	Se establecen y definen los términos y conceptos para el proyecto, orientando su desarrollo desde el análisis hasta la implementación.	Ninguno
Capítulo 4: Diseño e implementación	Se presenta la construcción del sistema desde su etapa inicial mediante diagramas y se presenta pseudocódigo implementado.	1, 2, 3 y 4
Capítulo 5: Despliegue	Se documenta el proceso de despliegue inicial de la aplicación.	4
Capítulo 6: Pruebas	En este capítulo se documentan las pruebas de usabilidad y aceptación, permitiendo una retroalimentación para etapas futuras (Capítulo 8).	6
Capítulo 7: Conclusiones	Se presentan las conclusiones del proyecto.	6
Capítulo 8: Trabajos Futuros	Se presentan propuestas para la expansión de la aplicación en el futuro.	Ninguno
Capítulo 9: Bibliografía	Se presentan las referencias y fuentes de información del proyecto.	Ninguno

Cuadro 1.2: Estructura de capítulos

## Capítulo 2

# Justificación y alcance del proyecto

### 2.1. Justificación

El desarrollo de una aplicación web para Beer & Coffee Company (BCC) surge de la necesidad de modernizar la interacción del cliente con la vasta gama de productos que ofrece el establecimiento. La abundancia de opciones disponibles actualmente genera incertidumbre entre los clientes y dificulta la toma de decisiones, fenómeno conocido como la paradoja de la elección.[2]

Por lo tanto, la aplicación web busca facilitar una navegación personalizada y sencilla a través del catálogo de productos, permitiendo realizar búsquedas, hacer pedidos y recibir recomendaciones basadas en filtrado colaborativo, este enfoque optimizaría la toma de decisiones, sino que también mejoraría la satisfacción del cliente al sentirse seguro de que el producto que escoja está bien detallado y es recomendado por muchos usuarios con gustos similares, permitiendo también la accesibilidad para elegir productos nuevos entre diversas y numerosas opciones. La implementación de este sistema requiere escalabilidad e independencia, por lo que una arquitectura de servicios independientes facilitará la integración sencilla de las funcionalidades que componen la aplicación y permitirá escalar el sistema conforme el negocio crezca.[9]

### 2.2. Alcance del Proyecto

Este proyecto consiste en el desarrollo de una aplicación web responsiva orientada al cliente. La aplicación permitirá a los clientes explorar el catálogo de productos, realizar pedidos, recibir recomendaciones personalizadas y utilizar un motor de búsqueda basado en procesamiento de lenguaje natural (PLN). Por otro lado, los administradores podrán gestionar los productos del catálogo mediante un sistema CRUD (crear, leer, actualizar y eliminar).

La solución adoptará una arquitectura de microservicios, lo que facilitará la escalabilidad y la flexibilidad para la integración de nuevas funcionalidades. Además, se proporcionará documentación técnica y un manual de usuario para asegurar una correcta implementación y uso del sistema.

### 2.2.1. Características Principales

- **Interfaz de Usuario Responsiva:**
  - Página inicial de acceso simple para usuarios con rol de cliente y administradores de BCC.
  - Página de catálogo de productos interactivo que permite explorar a detalle un producto específico.
  - Página que muestra el detalle de un producto específico, con la opción de añadir calificación, botón de agregar a la lista y botón de ir a la lista de pedidos.
  - Componentes de recomendaciones y popularidad.
  - Página de búsqueda que identifica los gustos del cliente.
  - Página para realizar pedidos.
  - Página de gestión del catálogo por parte del administrador de productos a través de un sistema CRUD.
  - Página de recepción de notificaciones.
  - Página de almacenamiento de pedidos.
- **Sistema de Recomendación:** Incluye un algoritmo de recomendación de filtrado colaborativo basado en datos guardados por las interacciones en tiempo real del cliente.
- **Arquitectura de Servicios:** Incluye servicios de recomendaciones, interacciones, popularidad, búsqueda, órdenes y gestión de productos.
- **Documentación del Proyecto:**
  - Documentación técnica de los componentes del proyecto.
  - Manual de usuario.

### 2.2.2. Exclusiones del Proyecto

- **Gestión de inventario detallada:** El proyecto no incluirá funcionalidades avanzadas para la gestión del inventario, solo agregación, actualización y eliminación de productos.
- **Funcionalidades de pago:** El proyecto no incluirá opciones de pago en línea, como tarjeta de crédito o pago virtual.
- **Integraciones externas:** No se incluirán integraciones con sistemas externos, como plataformas de envío o sistemas de gestión de inventarios.

### 2.2.3. Limitaciones y Suposiciones

- **Limitaciones:**
  - El sistema se diseñará únicamente para navegadores web y no incluirá una aplicación móvil nativa en esta fase.
  - La gestión de productos se limitará a operaciones CRUD, sin funciones avanzadas de inventario.
  - Las funcionalidades de pago en línea no se implementarán en esta versión del proyecto.
- **Suposiciones:**

- Se asume que los usuarios tendrán acceso a internet y un navegador web compatible.
- Se espera que el personal administrativo reciba la capacitación adecuada para utilizar el sistema.
- Se presupone que la infraestructura tecnológica actual del café-bar es suficiente para soportar la nueva aplicación.

## Capítulo 3

# Marco Referencial

### 3.1. Marco Teórico

#### 3.1.1. Sistemas de Recomendación

Los sistemas de recomendación (RS) son herramientas y técnicas de software que proporcionan sugerencias de elementos que probablemente sean de interés para un usuario en particular. Estas sugerencias generalmente se relacionan con procesos de toma de decisiones, como qué artículos elegir, qué música escuchar o qué noticias leer.[5]

Los RS son valiosos para ayudar a los usuarios en línea a enfrentar la sobrecarga de información y tomar mejores decisiones. Son una de las aplicaciones más populares de la inteligencia artificial, apoyando el descubrimiento de información en la web. Las recomendaciones suelen ser personalizadas y no personalizadas.

#### **Recomendaciones Personalizadas (RP)**

Son sugerencias de artículos adaptadas a las preferencias individuales del usuario, utilizando información explícita (calificaciones, comentarios) e implícita (comportamiento de navegación) para generar recomendaciones. En las RP se incluyen métodos de filtrado colaborativo que recomiendan ítems basados en la preferencia de usuarios similares y basados en contenido que recomiendan artículos similares a los que el usuario ha calificado positivamente.[5]

#### **Recomendaciones No Personalizadas (RNP)**

Son sugerencias aplicables a todos los usuarios, sin personalización, basadas en la popularidad general y tendencias actuales, sin considerar preferencias individuales. Algunos ejemplos son las listas de “Los más vendidos” o “Los más populares”. [5]

#### **Sistema de Recomendación Filtrado Colaborativo (CF)**

Métodos para hacer predicciones automáticas (filtrado) sobre los intereses de un usuario al recopilar preferencias o información de gusto de muchos usuarios. Incluyen:[5]

- **Basado en Usuarios:** Recomienda ítems basados en la similitud entre usuarios. Un sistema típico de este tipo busca usuarios que comparten el mismo patrón de calificación con el usuario activo y utiliza las calificaciones de estos usuarios similares para calcular una predicción para el usuario activo.[5]

- **Basado en Ítems:** “En lugar de medir la similitud entre usuarios, el sistema de recomendación basado en ítems calcula la similitud entre los ítems que los usuarios han calificado”. [5]

### 3.1.2. Sistema de Recomendación Basado en Contenido (CBF)

El filtrado basado en contenido, según describe Falk, utiliza características de los ítems para recomendar nuevos ítems similares a los que a un usuario le han gustado en el pasado. Este método depende en gran medida de la disponibilidad y precisión de los metadatos sobre los ítems. Los aspectos clave incluyen:

- **Creación de Perfil:** Se desarrolla un perfil de usuario basado en características de los ítems con los que el usuario ha interactuado.
- **Comparación de Ítems:** Se recomiendan ítems basados en su similitud con el perfil del usuario, utilizando métricas como la similitud coseno o la distancia euclidiana. [10]

### 3.1.3. Sistemas de Recomendación Híbridos

Falk discute los sistemas de recomendación híbridos, que combinan el filtrado colaborativo y el filtrado basado en contenido para mejorar el rendimiento de las recomendaciones y superar las limitaciones específicas de cada método. El enfoque híbrido puede variar en su implementación:

- **Ponderado:** Combinando predicciones de ambos modelos, colaborativos y basados en contenido, asignando pesos.
- **Conmutación:** El sistema alterna entre métodos colaborativos y basados en contenido, dependiendo de la situación o contexto.
- **Mixto:** Características o recomendaciones de ambos métodos, colaborativos y basados en contenido, se combinan en un solo modelo. [10]

### 3.1.4. Técnicas de Recomendación Basada en Grafos

Las técnicas de recomendación basada en grafos utilizan estructuras de grafos para representar relaciones entre usuarios y elementos. En Neo4j, una base de datos de grafos, estas técnicas aprovechan la capacidad del grafo para modelar y analizar conexiones complejas de manera eficiente. Los sistemas de recomendación basados en grafos permiten realizar recomendaciones en tiempo real al capturar instantáneamente nuevos intereses mostrados durante la visita de un usuario. Utilizando la biblioteca de Graph Data Science de Neo4j, se pueden implementar algoritmos avanzados como el filtrado colaborativo, que encuentra usuarios similares y recomienda elementos basados en las preferencias colectivas de dichos usuarios. [6][7]

### 3.1.5. Métricas de sistemas de recomendación basado en filtrado colaborativo (CF)

Se utilizan diversas métricas para medir la similitud entre usuarios o ítems y para evaluar la calidad de las recomendaciones. Algunas de las métricas más comunes incluyen:

- **Correlación de Pearson:** Mide la correlación lineal entre dos conjuntos de calificaciones (por ejemplo, entre dos usuarios o dos ítems). Es útil para determinar la similitud entre usuarios o ítems, ajustando las calificaciones para evitar diferencias individuales en la escala utilizada. [11, 12]

- **Cosine Similarity (Similitud Coseno):** Mide el coseno del ángulo entre dos vectores de calificaciones, donde un valor de 1 indica una similitud perfecta y un valor de 0 indica que no hay similitud. Es ampliamente utilizado en filtrado colaborativo para medir la similitud en un espacio vectorial. [13, 14]
- **Jaccard Similarity (Similitud de Jaccard):** Mide la similitud entre dos conjuntos como el tamaño de la intersección dividido por el tamaño de la unión. Es especialmente útil en sistemas donde las calificaciones son binarias. [15, 16]
- **Mean Squared Difference (Diferencia Cuadrática Media):** Calcula la diferencia promedio de las calificaciones entre dos usuarios o ítems y eleva al cuadrado las diferencias. Esta métrica es útil para medir la disimilitud entre usuarios o ítems. [17, 18]
- **Spearman Rank Correlation (Correlación de Rangos de Spearman):** Mide la correlación entre los rangos de las calificaciones de dos usuarios o ítems. Es una métrica no paramétrica que es útil cuando los datos no siguen una distribución normal. [19, 20]
- **Kullback-Leibler Divergence (Divergencia de Kullback-Leibler):** Mide la diferencia entre dos distribuciones de probabilidad. Es una métrica asimétrica que se utiliza para comparar la distribución de calificaciones entre dos usuarios o ítems. [21, 22]
- **Hamming Distance (Distancia de Hamming):** Mide la cantidad de posiciones en las que dos secuencias de igual longitud son diferentes. Se usa en sistemas donde las calificaciones son binarias, como en recomendaciones de "gustado." "no gustado". [23, 24]

### 3.1.6. Aplicaciones Web

Una aplicación web es un software que se ejecuta en el navegador web. Las empresas tienen que intercambiar información y proporcionar servicios de forma remota. Utilizan aplicaciones web para comunicarse con los clientes cuando lo necesiten y de una forma segura. Las funciones más comunes de los sitios web, como los carros de compra, la búsqueda y el filtrado de productos, la mensajería instantánea y los canales de noticias de las redes sociales, tienen el mismo diseño que las aplicaciones web. Le permiten acceder a funcionalidades complejas sin la necesidad de instalar o configurar un software.[8]

### 3.1.7. Servicios Web

Un servicio web es un sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable. Cuenta con una interfaz descrita en un formato procesable por un equipo informático (específicamente en WSDL), a través de la que es posible interactuar con el mismo mediante el intercambio de mensajes SOAP, típicamente transmitidos usando serialización XML sobre HTTP conjuntamente con otros estándares web.[25]

### 3.1.8. Arquitectura de Software

La arquitectura de software se refiere a la estructura fundamental de un sistema de software, comprendida por los componentes del software, las relaciones entre ellos y las propiedades de ambos. Se describe la arquitectura de software como "las decisiones más importantes que se deben tomar en el diseño de un sistema", que tienen un alto impacto en el sistema y sus partes interesadas. La arquitectura no solo abarca los componentes y sus interacciones, sino también, las decisiones que se tomaron para llegar a esa estructura y la lógica detrás de esas decisiones.[9] Algunas de estas arquitecturas son:



- **Arquitectura Monolítica:** La arquitectura monolítica es un estilo de arquitectura en el que una aplicación es construida como una única unidad indivisible. Todos los componentes del software están interconectados y desplegados como una sola aplicación, lo que implica que cualquier cambio o actualización requiere desplegar toda la aplicación nuevamente. Esta arquitectura puede ser más sencilla de desarrollar inicialmente, pero puede volverse difícil de mantener y escalar a medida que la aplicación crece en complejidad. Los sistemas monolíticos tienden a ser menos flexibles y más difíciles de escalar en comparación con las arquitecturas de microservicios.[9][26]
- **Arquitectura de Microservicios:** La arquitectura de microservicios es un enfoque para desarrollar una aplicación como un conjunto de pequeños servicios, cada uno ejecutándose en su propio proceso y comunicándose mediante mecanismos ligeros, a menudo una (application programming interface)API de recursos HTTP. Estos servicios están contruidos en torno a capacidades empresariales y son desplegables de manera independiente mediante maquinaria de despliegue automatizado. Según Martin Fowler, la arquitectura de microservicios reduce la gestión centralizada al mínimo y permite el uso de diferentes tecnologías de almacenamiento de datos y lenguajes de programación para diferentes servicios, facilitando así una mayor flexibilidad y escalabilidad.[27]

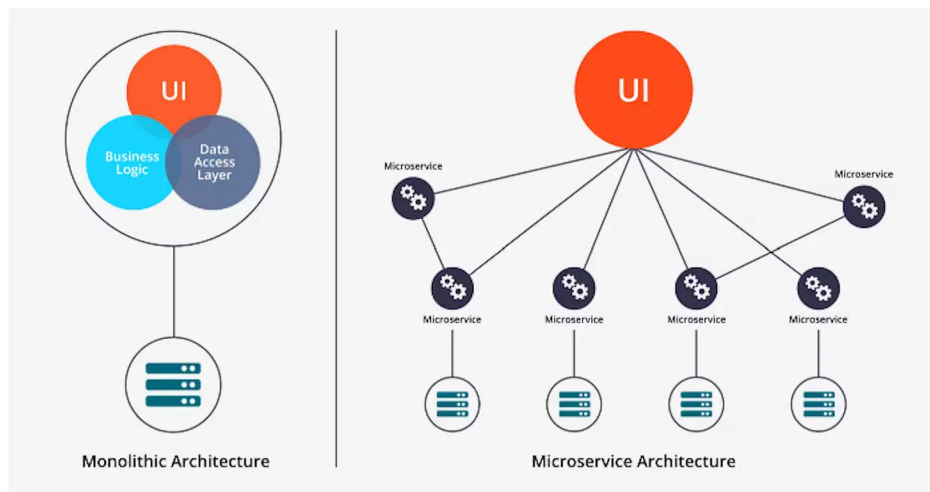


Figura 3.1: Arquitectura de microservicios y monolítica

### 3.1.9. Psicología del Consumidor

La paradoja de la elección es un concepto en psicología que postula que, aunque se cree que tener más opciones permite a los individuos tomar decisiones que se ajusten mejor a sus necesidades y deseos, tener demasiadas opciones puede provocar ansiedad y estrés, e incluso conducir a la inacción o a la insatisfacción. Esta teoría fue popularizada por el psicólogo Barry Schwartz, quien argumentó que “demasiadas opciones pueden paralizar al consumidor y hacer que se sienta menos satisfecho con su elección final”. [2]

## 3.2. Marco Conceptual

Teniendo las teorías establecidas, se establecen las definiciones fundamentales de los términos y conceptos clave que serán utilizados en el desarrollo de la aplicación web para Beer & Coffee Company. Estos conceptos son importantes para guiar todas las fases del proyecto, desde el análisis hasta la implementación, asegurando que cada aspecto del sistema esté claramente definido y alineado con los objetivos estratégicos.

### 3.2.1. Sistemas de recomendación

Conjunto de herramientas y técnicas que proporcionan sugerencias personalizadas y no personalizadas a los usuarios, basadas en algoritmos que predicen preferencias y comportamientos[5].

### 3.2.2. Arquitectura de microservicios

Arquitectura de software que organiza una aplicación como una colección de servicios pequeños, independientes y fácilmente gestionables, que interactúan mediante APIs de recursos HTTP[27].

### 3.2.3. Paradoja de Elección

Fenómeno psicológico donde un exceso de opciones puede llevar a la ansiedad y disminución de la satisfacción, potencialmente causando una parálisis en la toma de decisiones[2].

### 3.2.4. Experiencia del Usuario (UX)

La experiencia del usuario (UX) se refiere a la forma en que una persona interactúa con y percibe un producto, sistema o servicio. Según Don Norman, quien acuñó el término en la década de 1990, “La experiencia del usuario abarca todos los aspectos de la interacción del usuario final con la empresa, sus servicios y sus productos.”[28]. Esto incluye no solo la usabilidad y accesibilidad de un producto, sino también, aspectos emocionales y de satisfacción que el usuario experimenta durante y después de la interacción.

### 3.2.5. Recomendaciones basdas en grafos

En Neo4j, una base de datos de grafos, estas técnicas aprovechan la capacidad del grafo para modelar y analizar conexiones complejas de manera eficiente. Los sistemas de recomendación basados en grafos permiten realizar recomendaciones en tiempo real al capturar instantáneamente nuevos intereses mostrados durante la visita de un usuario[6][7].

### 3.2.6. Algoritmos de filtrado colaborativo basada en usuarios

Método de recomendación que predice las preferencias de un usuario basándose en las preferencias de usuarios similares, utilizando la teoría de grafos para mapear relaciones entre usuarios y productos. Recomienda ítems basados en la similitud entre usuarios. Un sistema típico de este tipo busca usuarios que comparten la misma calificación de patrones con el usuario activo y utiliza las calificaciones de estos usuarios ‘similares’ para calcular una predicción para el usuario activo[5].

## Capítulo 4

# Diseño e implementación

### 4.1. Diseño e Implementación de la Interfaz

El diseño e implementación del frontend comenzó con la creación de wireframes en Figma[29]. Esta herramienta permitió delinear de manera precisa y estructurada el diseño y la funcionalidad de la interfaz de usuario para la página destinada al cliente. El wireframe sirvió como un mapa visual preliminar, facilitando la planificación de los elementos y la disposición general de la página.

El objetivo de los wireframes era proporcionar una representación inicial del aspecto y la experiencia del usuario de la aplicación antes de comenzar su desarrollo real. Esto permitió obtener una idea preliminar de cómo se estructuraría la aplicación durante el desarrollo.

Posteriormente, con React[30], se empezó la construcción de la interfaz debido a su eficiencia y flexibilidad para crear aplicaciones web dinámicas y componentes reutilizables. La estructura (JavaScript Syntax eXtension) JSX proporcionó el esqueleto de la aplicación, mientras que (Cascading Style Sheets) CSS se utilizó para estilizar los componentes, asegurando una apariencia moderna y cohesiva.

Cabe destacar que la interfaz presentada es la primera versión y está diseñada de manera que pueda ser mejorada en el futuro, ajustándose a las necesidades y preferencias de los usuarios para ofrecer una experiencia más satisfactoria.

#### 4.1.1. Prototipo e interfaces de usuario

##### Prototipo e interfaz Login

###### **Seleccionar Tipo de Usuario:**

- Cliente: Haz clic en el botón con el icono de usuario (Cliente).
- Admin: Haz clic en el botón con el icono de escudo de usuario (Admin).

###### **Ingresar Credenciales:**

- Cliente: Introduce tu nickname en el campo “Nickname”.
- Admin: Introduce tu nickname en el campo “Nickname” y tu contraseña en el campo “Contraseña”.

###### **Iniciar Sesión:**

- Haz clic en el botón “Ingresar” para enviar tus credenciales.

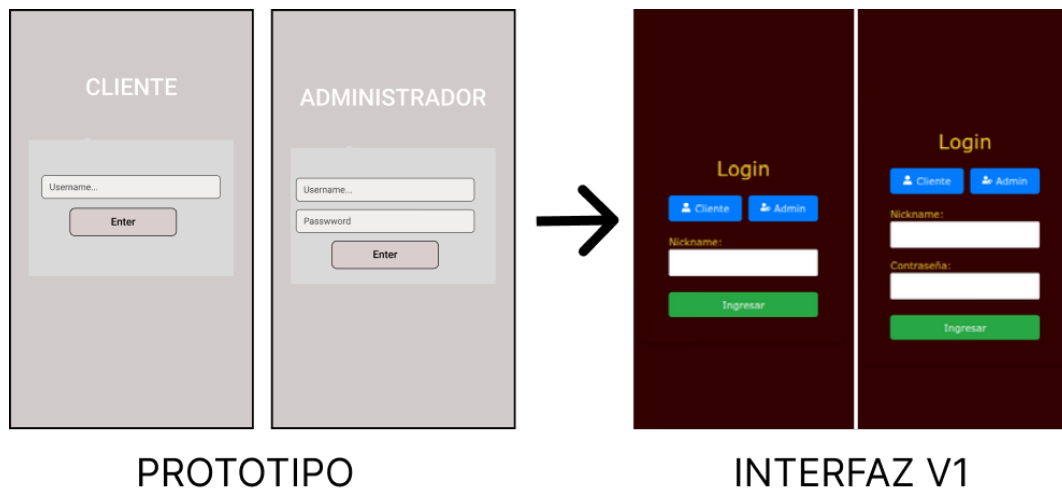


Figura 4.1: Prototipo e interfaz login de usuario para cliente y administrador

### Prototipo e interfaz inicio cliente

#### Catálogo de Productos:

- Ver Productos por Categoría: Los productos están organizados en categorías con un botón “Ver más” para explorar la categoría completa.
- Explorar Productos: Utiliza un carrusel (Swiper) para deslizarse a través de los productos.
- Detalles del Producto: Haz clic en cualquier producto para ver detalles adicionales.

#### Recomendaciones Personalizadas:

- Muestra productos recomendados basados en las interacciones previas del usuario.

#### Productos Populares:

- Presenta los productos más comprados por otros usuarios.

#### Guardar Interacción:

- Las interacciones con productos y categorías se guardan sin enterarse para mejorar las recomendaciones futuras.



Figura 4.2: Prototipo e interfaz inicio de la página catálogo de productos

### Prototipo e interfaz página de búsqueda cliente

Permite buscar productos específicos en la base de datos.

#### Funcionalidades:

- **Buscar Productos:** Introduce términos de búsqueda en la barra de búsqueda para encontrar productos específicos.
- **Visualización de Resultados:** Muestra los productos encontrados con su imagen, nombre y precio.

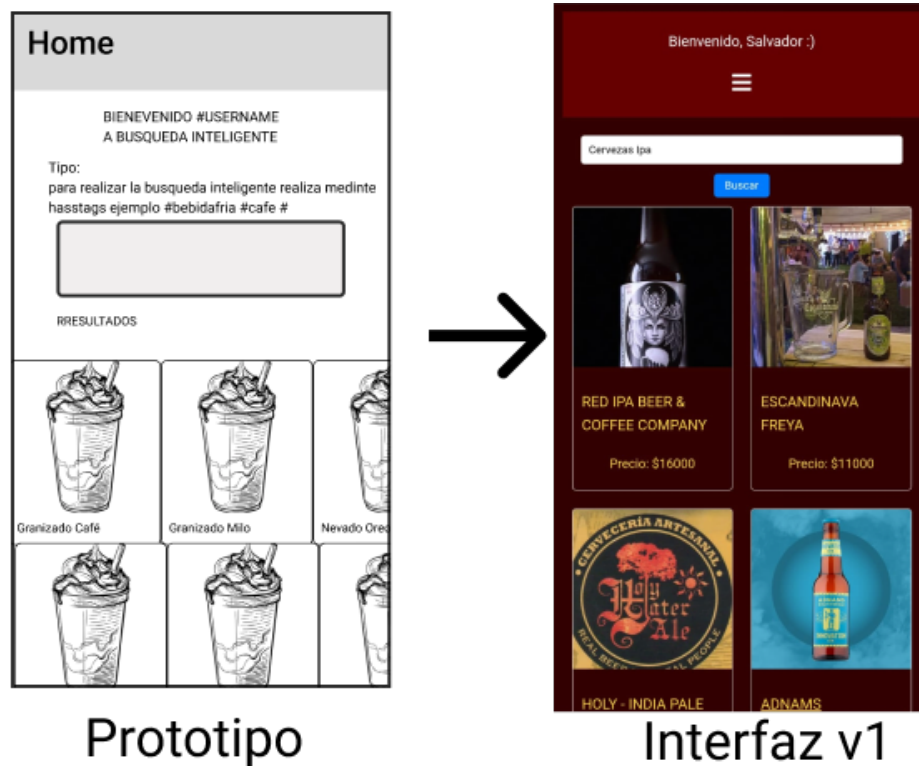


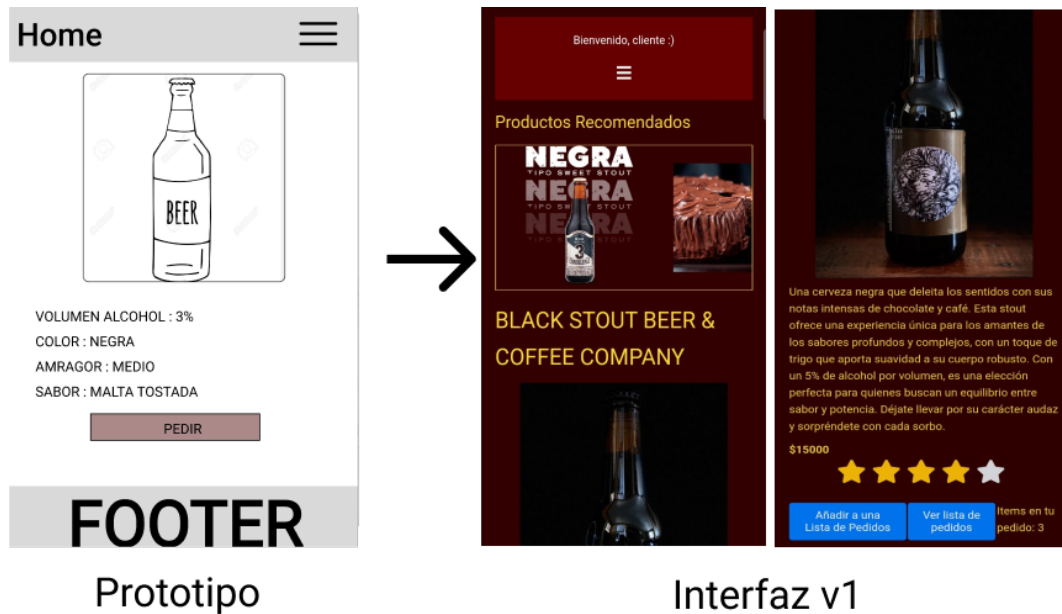
Figura 4.3: Prototipo e interfaz inicio de cliente

### Prototipo e interfaz detalles del producto

Muestra información detallada del producto seleccionado.

#### Funcionalidades:

- Visualización de Detalles del Producto: Muestra la imagen, descripción y precio del producto.
- Valoración del Producto: Permite valorar el producto con un sistema de estrellas (1-5).
- Añadir a la Lista de Pedidos: Permite añadir el producto a tu lista de pedidos y mirar la lista de pedidos.



Prototipo

Interfaz v1

Figura 4.4: Prototipo e interfaz para cliente de detalles del producto

### Prototipo e interfaz lista de pedidos

**Gestiona la lista de productos añadidos a tu pedido.**

**Funcionalidades:**

- Visualización y Gestión de la Lista de Pedidos: Muestra los productos añadidos a tu lista de pedidos, agrupados por producto y cantidad.
- Permite reducir la cantidad de un producto o eliminarlo de la lista.
- Realizar Pedido: Calcula el precio total de los productos en tu lista. Permite introducir un número de mesa opcional y realizar el pedido.

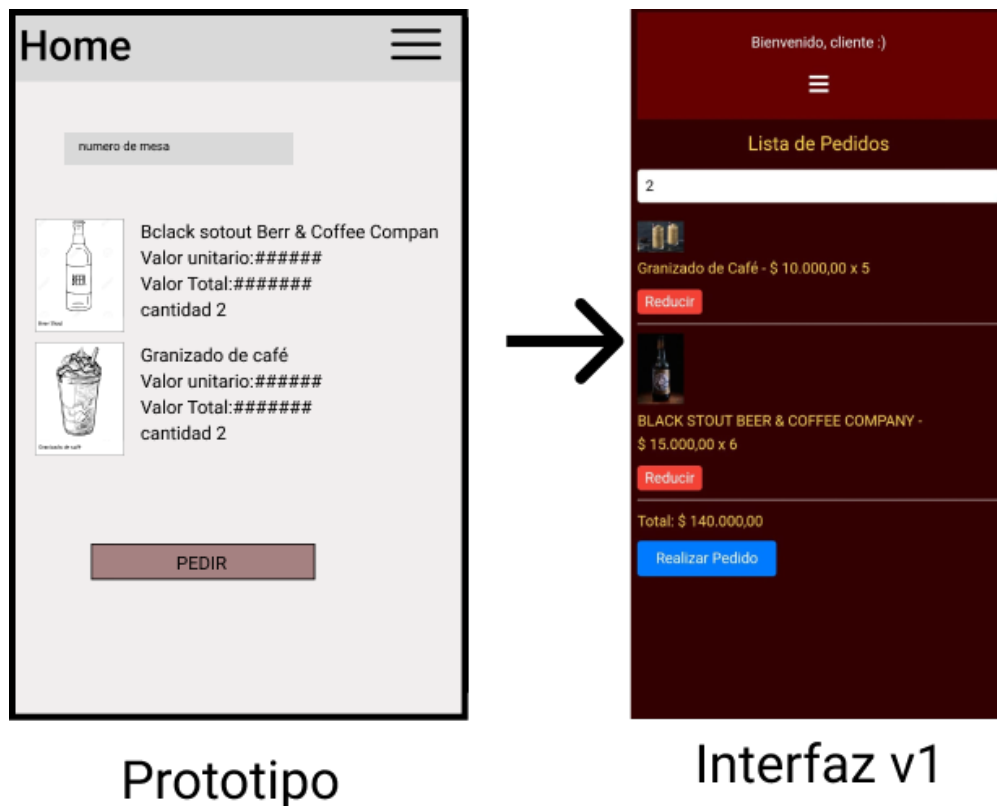


Figura 4.5: Prototipo e interfaz de lista de ordenes cliente

### Prototipo e interfaz páginas del administrador

#### Página de Inicio del Administrador (HomePageAdmin):

- La página de inicio del administrador presenta enlaces rápidos a las secciones de gestión de productos, notificaciones de pedidos y el historial de ventas.

#### Permite al administrador gestionar los productos de la tienda.

##### Funcionalidades:

- Visualización de Productos: Muestra una lista de productos con sus detalles (nombre, categoría, precio, descripción).
- Añadir/Editar Productos: Formulario para añadir un nuevo producto o editar uno existente.
- Campos para nombre, categoría, precio, descripción y la URL de la imagen.
- Eliminar Productos: Botón para eliminar un producto existente.



**Página de Notificaciones de Pedidos (OrderNotificationsPage):**

- Permite al administrador ver las notificaciones de nuevos pedidos y gestionarlos.
- Visualización de Notificaciones: Muestra una lista de notificaciones de nuevos pedidos con detalles (ID del pedido, número de mesa, cliente, fecha del pedido, total).
- Interacción con Notificaciones: Marcar notificaciones como leídas al hacer clic en ellas.
- Eliminar notificaciones.
- Guardar pedidos en el historial de ventas.

**Página de Historial de Ventas (SalesHistoryPage):**

- Permite al administrador ver el historial de ventas agrupado por fecha.
- Visualización del Historial de Ventas: Muestra el historial de ventas con detalles de cada pedido (ID del pedido, número de mesa, fecha del pedido, total, items del pedido).
- Agrupación por Fecha: Los pedidos se agrupan por fecha y se muestra el total de ventas por día.

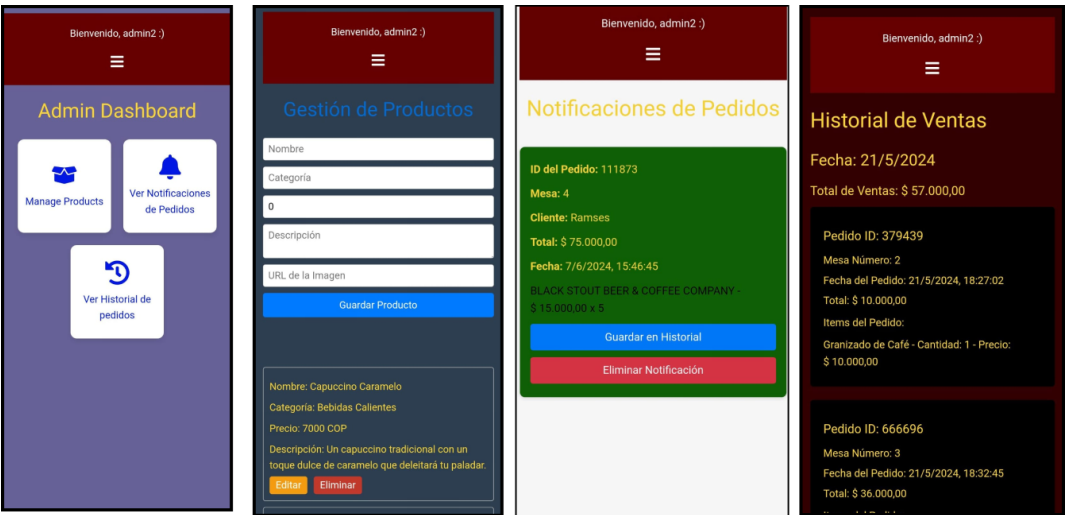


Figura 4.6: Panel de administración y páginas

En la figura 4.5 las páginas del administrador se implementan directamente a medida que se desarrolla la aplicación, sin un prototipo inicial en Figma. Este enfoque permitió realizar ajustes y mejoras sobre la marcha, basados en la funcionalidad y las necesidades específicas del sistema.

## 4.2. Arquitectura de la Aplicación

### 4.2.1. Introducción

El diseño e implementación del *backend* se basa en un enfoque de servicio, que consiste en la descomposición de la aplicación en componentes autónomos que ejecutan funciones específicas y se comunican a través de *APIs*. Este diseño permite que cada servicio funcione de manera independiente, facilitando actualizaciones y escalabilidad sin afectar otros componentes del sistema.

- **Independencia de Servicios:** Cada servicio opera de manera independiente, lo que permite actualizaciones o modificaciones sin repercutir en otros componentes del sistema.
- **Escalabilidad:** Los servicios permiten una escalabilidad independiente basada en la demanda de cada uno, optimizando el uso de recursos y mejorando el rendimiento global.
- **Diversidad Tecnológica:** Cada servicio puede ser desarrollado, desplegado y gestionado utilizando las tecnologías más adecuadas para sus necesidades específicas.
- **Gestión Centralizada:** *Nginx*, como *API Gateway*, centraliza la gestión del tráfico y facilita el balanceo de carga.

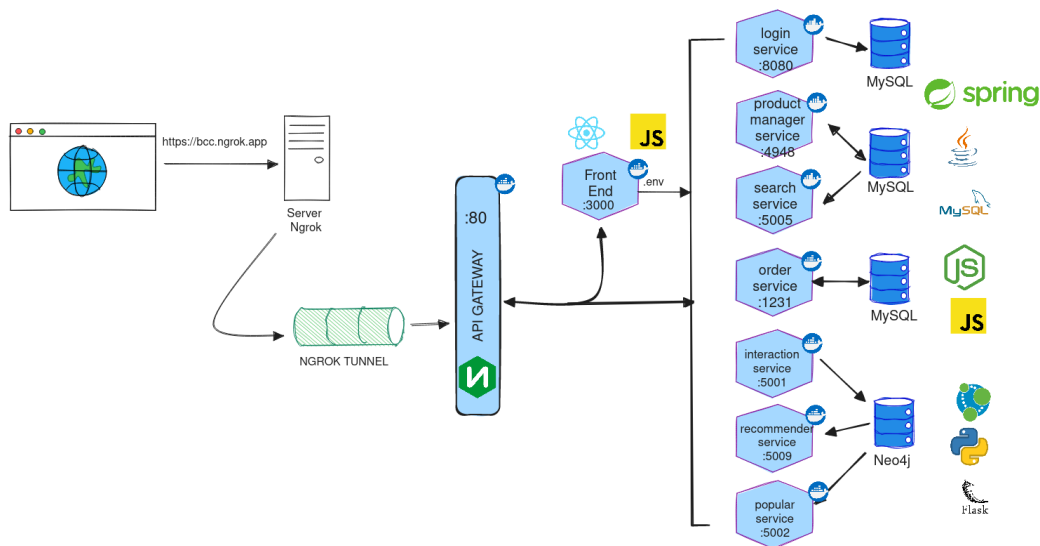


Figura 4.7: Arquitectura general de la aplicación

#### 4.2.2. Proceso de Funcionamiento de la Aplicación

El flujo operativo desde el acceso del usuario hasta el procesamiento de solicitudes por parte de los microservicios.

1. **Acceso a la Aplicación:** El usuario accede mediante un navegador al URL proporcionado por Ngrok.
2. **Procesamiento de la Solicitud a través de Ngrok:** Ngrok recibe y redirige la solicitud al API Gateway.
3. **Redirección por el API Gateway:** Nginx redirige las solicitudes al frontend o a los servicios correspondientes según la naturaleza de la solicitud.
4. **Gestión de la Interfaz de Usuario:** El frontend, desarrollado en React, gestiona la interfaz y envía solicitudes a los servicios adecuados.

### Servicios Implicados

Los microservicios clave y sus roles en la arquitectura.

- **Login Service:** Autentica usuarios y gestiona datos de sesión.
- **Product Manager Service:** Gestiona información y operaciones CRUD de productos.
- **Search Service:** Realiza búsquedas de productos y retorna resultados relevantes.
- **Order Service:** Administra los pedidos de los usuarios.
- **Interaction Service:** Registra interacciones de los usuarios con los productos.
- **Recommender Service:** Genera recomendaciones personalizadas basadas en interacciones.
- **Popular Service:** Escoge las interacciones mas frecuentes y se calcula el promedio mostrando los productos más populares.

### Bases de Datos Involucradas

- **MySQL:** Utilizada para almacenar y gestionar datos en los servicios de Login, Product Manager, Search, y Order.
- **Neo4j:** Usada para analizar y almacenar interacciones y patrones de comportamiento en los servicios de Interacciones, Recomendaciones y Popularidad.

El proceso de funcionamiento se puede visualizar en el diagrama de secuencias adjunto, que muestra la interacción entre los diferentes componentes y servicios para mas detalle vease en los anexos Capitulo 9.

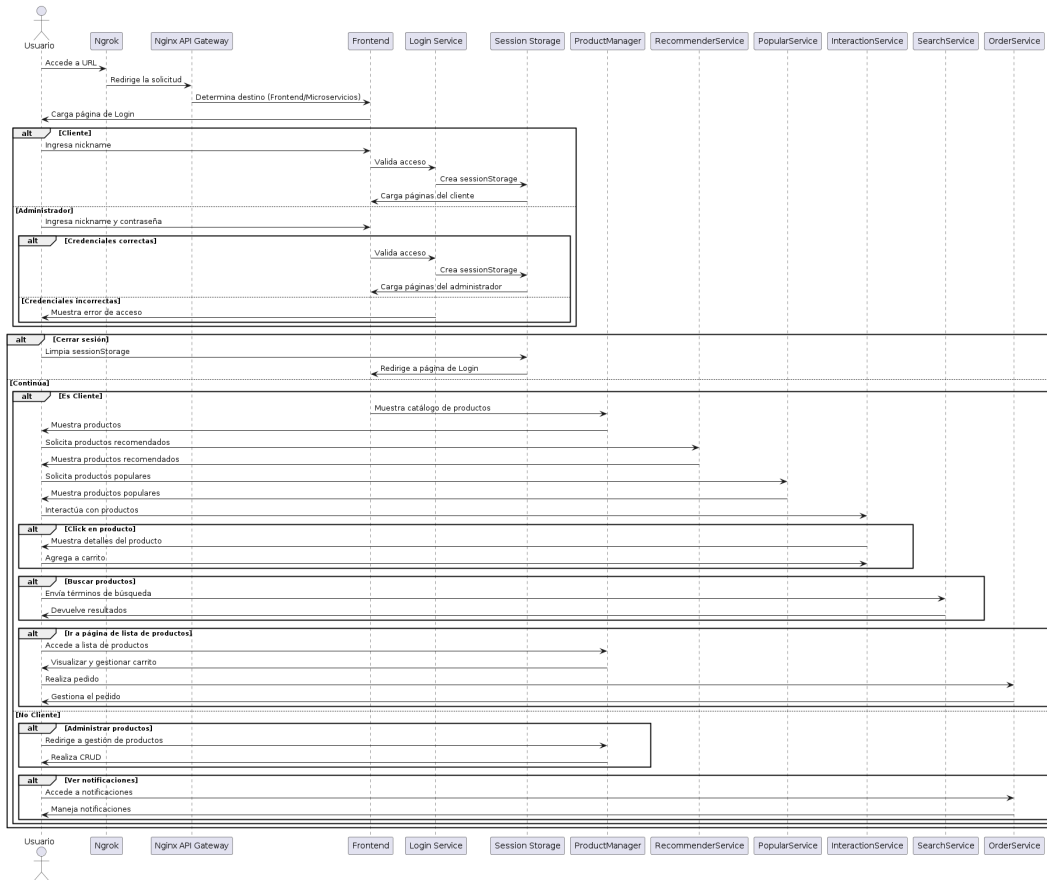


Figura 4.8: Diagrama de secuencias de la aplicación

## 4.3. Diseño e implementación del sistema de recomendación

### 4.3.1. Elección del Sistema de Recomendación de Filtrado Colaborativo

La elección de un algoritmo de recomendación basado en filtrado colaborativo para Beer & Coffee Company se fundamenta en que las recomendaciones se centran en la calidad del producto más que en las preferencias individuales, lo que fomenta el descubrimiento de nuevos productos. En lugar de limitarse a las características explícitas de los productos, como haría un sistema basado en contenido, el filtrado colaborativo utiliza las interacciones de los usuarios para identificar y recomendar productos que han sido bien valorados por otros con gustos similares. Por ejemplo, si un cliente elige una cerveza, un algoritmo basado en contenido tendería a recomendar únicamente cervezas con características similares, limitando la exploración del usuario a nuevos tipos de cerveza que podrían gustarle. En contraste, el filtrado colaborativo permite descubrir productos diversos basados en las preferencias de usuarios similares.

Un desafío del filtrado colaborativo es el problema de arranque en frío [31], donde la memoria de la base de datos para los nuevos usuarios tienen poca o ninguna información de interacción, dificultando la generación de recomendaciones precisas. Sin embargo, este problema se aborda presentando productos populares (recomendación no personalizada) desde el principio. En todo caso, el algoritmo de filtrado colaborativo basado en usuario expande el espectro de recomendaciones más allá de las preferencias explícitas, ofreciendo una experiencia de descubrimiento enriquecedora y diversa.

### 4.3.2. Recomendaciones en Tiempo Real

El sistema de recomendación ha sido diseñado para proporcionar recomendaciones en tiempo real, lo que significa que las sugerencias se generan y presentan al usuario de manera instantánea, basándose en sus interacciones más recientes. Esto es crucial para mantener la relevancia y la personalización de las recomendaciones, adaptándose rápidamente a los cambios en los intereses del usuario. Las recomendaciones empiezan a aparecer cuando el usuario comienza a interactuar con la aplicación. Inicialmente, se muestran productos populares para superar el problema de arranque en frío y ofrecer al usuario una base de exploración.

### 4.3.3. Funcionamiento de los grafos en el sistema de recomendación

El sistema de recomendaciones utiliza una base de datos orientada a grafos, como Neo4j, para registrar y analizar las interacciones de los usuarios con los productos como se puede ver en el modelo de grafos, figura 4.10. En un grafo, los nodos representan tanto a los usuarios como a los productos, mientras que las relaciones entre ellos representan las interacciones, como compras o valoraciones. Este modelo permite una representación de las conexiones en los datos de recomendación.

Mediante consultas Cypher implementadas en una API en Python, el servicio de interacciones se encarga de registrar los nodos y las interacciones en el modelo de grafos. Los servicios de popularidad y recomendaciones analizan estos datos para presentar recomendaciones personalizadas y no personalizadas[32]. Por ejemplo, un algoritmo de filtrado colaborativo basado en grafos puede utilizar el algoritmo *El Coeficiente de Correlacion de Pearson* para identificar usuarios similares basándose en sus patrones de interacción con productos. Este enfoque permite aprovechar tanto las relaciones directas como las indirectas en el grafo para generar recomendaciones más precisas y diversificadas[33] [34] .

La arquitectura del sistema de recomendación incluye varios servicios necesarios pero también se resaltan los servicios principales. La integración de estos servicios permite una recopilación y análisis de datos eficiente, asegurando que las recomendaciones se actualicen y ajusten continuamente según las interacciones en tiempo real de los usuarios con el sistema.

- **LoginService:** Servicio que gestiona la autenticación de los usuarios y crea un nodo de usuario con datos como el nickname y el ID del usuario. Estos datos se utilizan para personalizar la experiencia del usuario y se gestionan en el frontend mediante React Context o Hooks, lo que permite mantener un estado de autenticación consistente a lo largo de la sesión.
- **ProductManager:** Administra la información de los productos disponibles en el catálogo. Este servicio interactúa directamente con el frontend para proporcionar y actualizar los datos de los productos, que se utilizan para crear y actualizar nodos en la base de datos Neo4j.

Servicios principales que se enfocan en la obtención de las recomendaciones:

- **InteractionService:** Esencial para el sistema de recomendaciones, este servicio registra cada interacción de los usuarios con los productos, como visualizaciones, clics y compras. Estos datos son almacenados en Neo4j y se utilizan para construir un grafo detallado de interacciones que ayuda a entender mejor las preferencias y comportamientos de los usuarios.
- **RecommenderService:** Utiliza un algoritmo de filtrado colaborativo basado en usuarios para ofrecer recomendaciones personalizadas. Este servicio procesa los grafos de interacciones almacenados en Neo4j para identificar patrones y similitudes entre los usuarios, permitiendo sugerencias más precisas y relevantes basadas en el comportamiento previo.

- **PopularService:** Proporciona recomendaciones basadas en la popularidad de los productos. Este servicio analiza las interacciones globales de todos los usuarios para determinar cuáles productos son los más populares, ofreciendo así recomendaciones no personalizadas que son útiles para nuevos usuarios o aquellos con ningún dato de interacción.

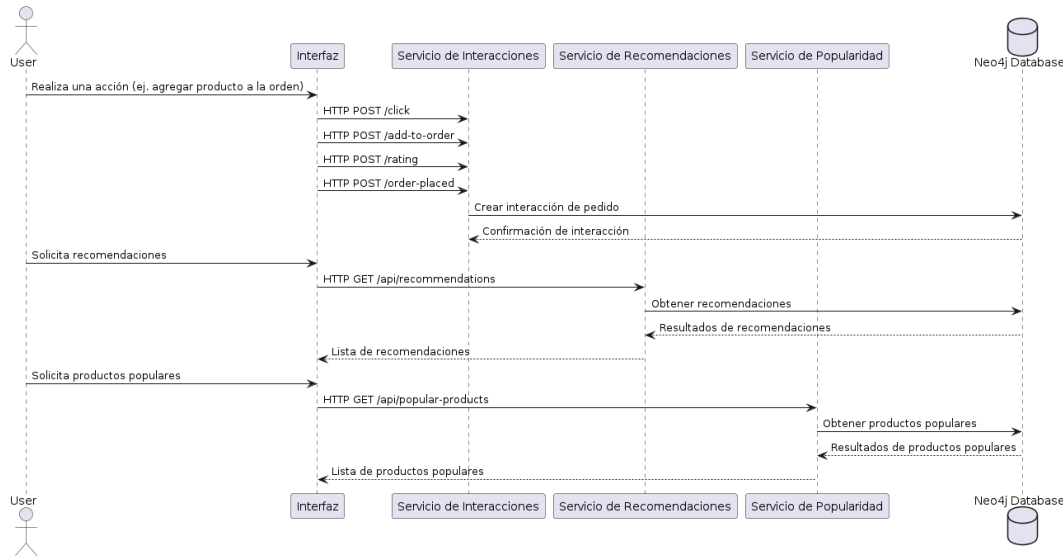


Figura 4.9: Diagrama de secuencias del diseño sistema de recomendaciones

#### 4.3.4. Modelo de base de datos NEO4J

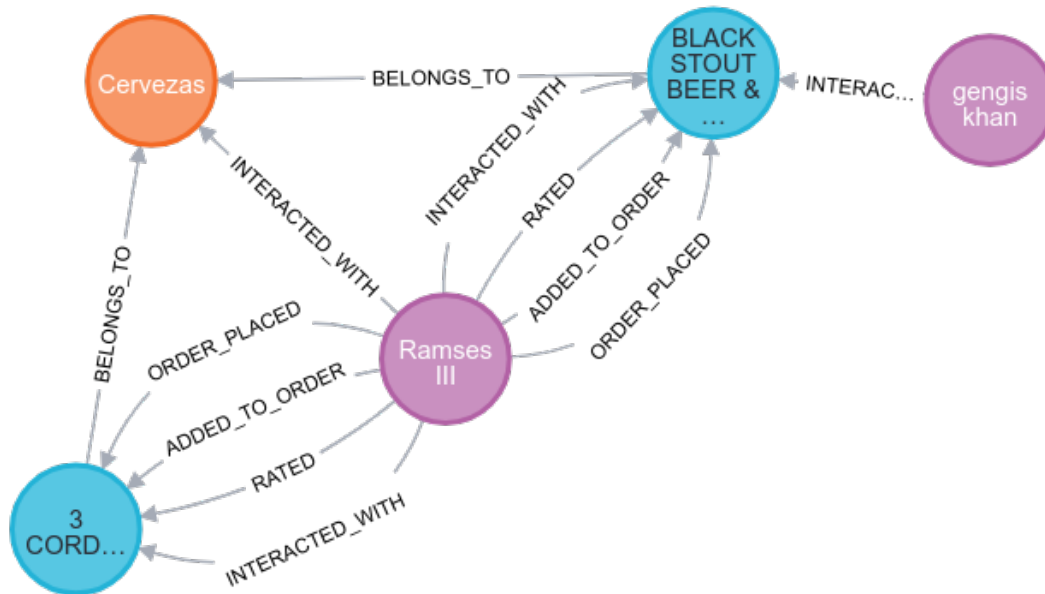


Figura 4.10: Modelo de la base de datos con nodos e interacciones guardadas en neo4j

La figura 4.10 muestra un modelo de base de datos en Neo4j, donde se refleja cómo se agrupan los productos (nodos azules) en categorías (nodos naranjas) y se destacan las acciones realizadas por los usuarios (nodos morados), proporcionando una representación visual de las relaciones y actividades dentro de la

plataforma. Estas acciones incluyen interacciones con los productos, calificaciones, adiciones a órdenes y colocaciones de órdenes, lo que permite entender el comportamiento y las preferencias de los usuarios en relación con los productos.

Composición del grafo más detallada:

- **Nodos de Usuario:** Representan a los individuos que interactúan con el sistema, realizan pedidos y reciben recomendaciones. Cada nodo de usuario almacena atributos como el ID del usuario y el nombre. Ejemplos de estos nodos incluyen “*Ramses III*” y “*Gengis Khan*”.
- **Nodos de Producto:** Representan los diversos productos disponibles en el sistema, como diferentes tipos de cervezas y café. Cada nodo de producto almacena atributos como el ID del producto y el nombre del producto. Ejemplos de estos nodos incluyen “*BLACK STOUT BCC*”, “*3 CORDILLERAS NEGRA*” y “*Cervezas*”.
- **Nodos de Categoría:** Representan las categorías a las que pertenecen los productos. Cada nodo de categoría almacena atributos como el ID de la categoría y el nombre de la categoría. Ejemplos de estos nodos incluye ejemplo ‘Cervezas’

Las relaciones entre los nodos son fundamentales para capturar las dinámicas del sistema. A continuación se describen las principales relaciones utilizadas:

#### ■ INTERACTS\_WITH

- **Descripción:** Indica una interacción general entre un usuario y un producto, capturando actividades como ver o explorar un producto.
- **Ejemplo:** Ramses III y Gengis khan → INTERACTS\_WITH → BLACK STOUT BCC (donde sería un nodo representando por el click de un producto específico)

#### ■ RATED

- **Descripción:** Captura la acción de un usuario al calificar un producto, incluyendo detalles como la puntuación dada por el usuario.
- **Ejemplo:** Ramses III → RATED → BLACK STOUT BCC y 3 CORDILLERA NEGRA (donde sería un nodo representando por la calificación de producto específico)

#### ■ ORDER\_PLACED

- **Descripción:** Indica que un usuario ha realizado un pedido que incluye ciertos productos.
- **Ejemplo:** Ramses III → ORDER\_PLACED → BLACK STOUT BCC y 3 CORDILLERA NEGRA (donde sería un nodo representando por la interacción de pedido específico)

#### ■ ADDED\_TO\_ORDER

- **Descripción:** Muestra que un producto ha sido añadido a un pedido específico.
- **Ejemplo:** Ramses III → ADDED\_TO\_ORDER → BLACK STOUT BCC

#### ■ BELONGS\_TO

- **Descripción:** Indica la asociación de productos con sus respectivas categorías, organizando los productos de manera estructurada.

- **Ejemplo:** BLACK STOUT BEER & COFFE COMPNAY y 3 CORDILLERA NEGRA → BELONGS\_TO → Cervezas

#### ■ CATEGORY\_INTERACTS\_WITH

- **Descripción:** Indica una interacción general entre un usuario y una categoría, capturando actividades como ver o explorar una categoría.
- **Ejemplo:** Ramses III → CATEGORY\_INTERACTS\_WITH → Cervezas

Apoyando la capacidad del motor de recomendaciones para proporcionar sugerencias personalizadas y relevantes a los usuarios según su historial de interacciones y preferencias. Las relaciones como INTERACTS\_WITH, RATED, ORDER\_PLACED y ADDED\_TO\_ORDER capturan diferentes tipos de interacciones entre los usuarios y los productos, proporcionando una base para generar recomendaciones. Además, las relaciones BELONGS\_TO y CATEGORY\_INTERACTS\_WITH organizan los productos en categorías, lo cual es clave para ofrecer recomendaciones específicas para el usuario.

### 4.3.5. Implementacion de algoritmos

Se implementa algoritmos específicos para registrar las interacciones de los usuarios con los productos y categorías en la base de datos Neo4j. Estos algoritmos son fundamentales para recopilar y gestionar las interacciones, que luego son utilizadas por los sistemas de recomendación basados en filtrado colaborativo *El Coeficiente de Correlacion de Pearson* y de productos populares, aprovechando los datos almacenados de las interacciones.

A continuación, se presenta la descripción detallada de las consultas utilizadas en la implementación de los algoritmos.

#### Algoritmos de interacciones

##### Guardar Interacción de Producto

Esta consulta guarda la interacción de un usuario con un producto específico. Se crea o actualiza la relación entre el usuario y el producto, registrando la cantidad de veces que el usuario ha interactuado con dicho producto.

---

#### Algorithm 1 Guardar Interacción de Producto

---

**Require:** *user\_id*, *user\_nickname*, *product\_id*, *product\_name*, *product\_category*, *timestamp*

- 1: **MERGE** (u:User {id: *user\_id*})
  - 2: **ON CREATE SET** u.nickname = *user\_nickname*
  - 3: **MERGE** (p:Product {id: *product\_id*})
  - 4: **ON CREATE SET** p.name = *product\_name*
  - 5: **MERGE** (c:Category {name: *product\_category*})
  - 6: **MERGE** (p)-[:BELONGS\_TO]->(c)
  - 7: **MERGE** (u)-[:INTERACTED\_WITH]->(p)
  - 8: **ON CREATE SET** r.count = 1, r.timestamp = *timestamp*
  - 9: **ON MATCH SET** r.count = r.count + 1, r.timestamp = *timestamp*
- 

##### Guardar Interacción de Categoría

Esta consulta registra la interacción de un usuario con una categoría específica, actualizando la relación entre el usuario y la categoría.



---

**Algorithm 2** Guardar Interacción de Categoría

---

**Require:** *user\_id*, *user\_nickname*, *category*, *timestamp*

- 1: **MERGE** (u:User {id: *user\_id*})
  - 2: **ON CREATE SET** u.nickname = *user\_nickname*
  - 3: **ON MATCH SET** u.nickname = *user\_nickname*
  - 4: **MERGE** (c:Category {name: *category*})
  - 5: **MERGE** (u)-[r:INTERACTED\_WITH]-i(c)
  - 6: **ON CREATE SET** r.count = 1, r.timestamp = *timestamp*
  - 7: **ON MATCH SET** r.count = r.count + 1, r.timestamp = *timestamp*
- 

**Guardar Calificación**

Esta consulta se encarga de registrar la calificación de un producto por parte de un usuario, incluyendo el producto y la categoría.

---

**Algorithm 3** Guardar Calificación

---

**Require:** *user\_id*, *user\_nickname*, *product\_id*, *product\_name*, *product\_category*, *rating*, *timestamp*

- 1: **MERGE** (u:User {id: *user\_id*})
  - 2: **ON CREATE SET** u.nickname = *user\_nickname*
  - 3: **ON MATCH SET** u.nickname = *user\_nickname*
  - 4: **MERGE** (c:Category {name: *product\_category*})
  - 5: **MERGE** (p:Product {id: *product\_id*})
  - 6: **ON CREATE SET** p.name = *product\_name*, p.category = *product\_category*
  - 7: **ON MATCH SET** p.name = **COALESCE**(*product\_name*, p.name), p.category = **COALESCE**(*product\_category*, p.category)
  - 8: **MERGE** (p)-[:BELONGS\_TO]-i(c)
  - 9: **MERGE** (u)-[:RATED {score: *rating*, timestamp: *timestamp*}] -i(p)
- 

**Agregar a Pedido**

Esta consulta registra la acción de agregar un producto a la lista de pedidos del usuario, actualizando la relación entre el usuario y el producto.

---

**Algorithm 4** Agregar Producto a Pedido

---

**Require:** *user\_id*, *product\_id*, *timestamp*

- 1: **MATCH** (u:User {id: *user\_id*})
  - 2: **MATCH** (p:Product {id: *product\_id*})
  - 3: **MERGE** (u)-[r:ADDED\_TO\_ORDER]-i(p)
  - 4: **ON CREATE SET** r.count = 1, r.timestamp = *timestamp*
  - 5: **ON MATCH SET** r.count = r.count + 1, r.timestamp = *timestamp*
- 

**Registrar Pedido**

Esta consulta finaliza el pedido de un producto, combinando la información de las interacciones previas y actualizando la relación en la base de datos.

**Algorithm 5** Registrar Pedido**Require:** *user\_id*, *product\_id*, *timestamp*

- 1: **MATCH** (u:User {id: *user\_id*})-[r:ADDED\_TO\_ORDER]-i(p:Product {id: *product\_id*})
- 2: **WITH** u, p, r.count **AS** added\_count
- 3: **MERGE** (u)-[orderPlaced:ORDER\_PLACED]-i(p)
- 4: **ON CREATE SET** orderPlaced.count = added\_count, orderPlaced.timestamp = *timestamp*
- 5: **ON MATCH SET** orderPlaced.count = orderPlaced.count + added\_count, orderPlaced.timestamp = *timestamp*

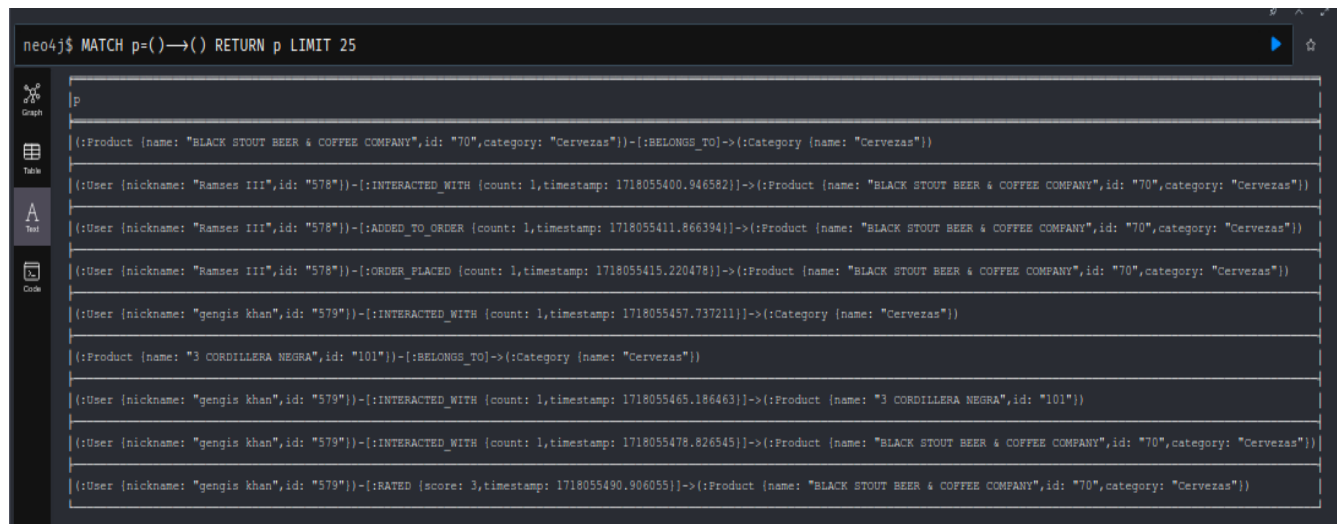
**Resultado de los algoritmos de Interacciones**

Figura 4.11: Interacciones almacenadas en la base de datos Neo4j

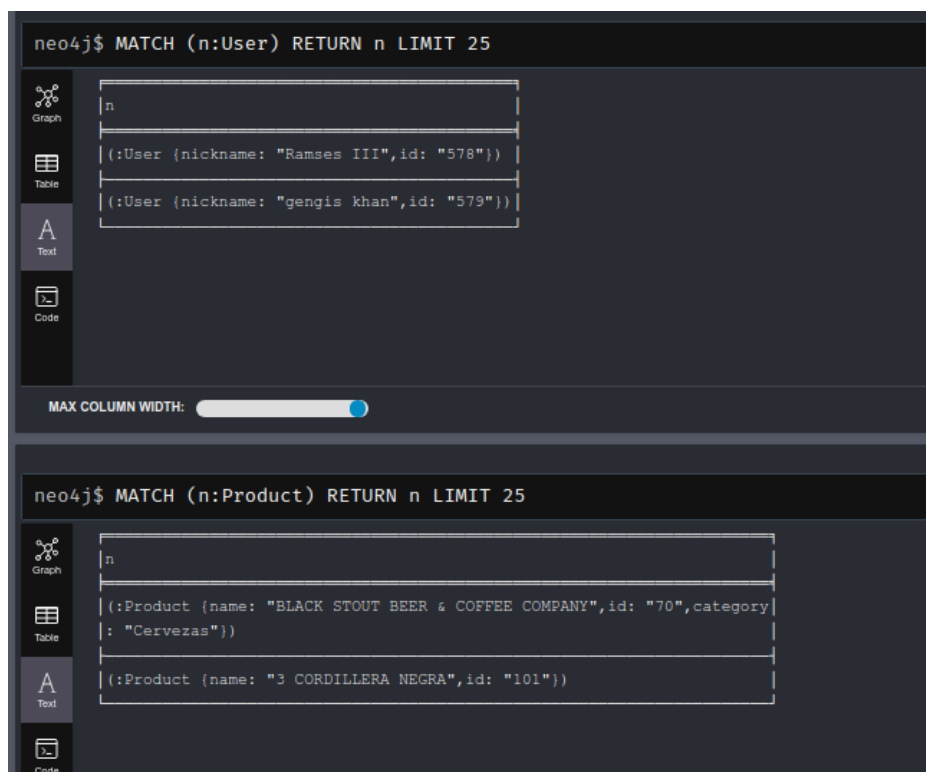


Figura 4.12: Nodo usuario y producto guardados en Neo4j

### Algoritmo de Recomendación Filtrado Colaborativo Basado en Usuarios

Se presenta el servicio RecommenderService, el cual, utiliza una adaptación a la fórmula del Coeficiente de Correlación de Pearson como se muestra en la figura 4.13. En este algoritmo, se identifica al usuario objetivo y se recuperan sus interacciones con los productos, asignando un peso a cada tipo de interacción: calificación (utilizando la calificación real), orden realizada (peso de 4), añadido a la lista de pedidos (peso de 3) e interacción simple (peso de 1). Luego, se calculan las similitudes entre el usuario objetivo y otros usuarios en función de estas interacciones ponderadas utilizando la fórmula de correlación de Pearson, que considera tanto las medias de las interacciones como las diferencias respecto a estas medias. Basándose en estas similitudes, se pueden recomendar productos que hayan sido altamente valorados o frecuentemente interactuados por los usuarios más similares.[34]

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

Figura 4.13: Fórmula para calcular la correlacion de Pearson.[35]

Finalmente, se recomiendan productos con los que estos usuarios similares han interactuado, pero que el usuario objetivo aún no ha explorado. Se calcula la frecuencia de recomendación para cada producto, se normaliza y se determina el porcentaje de recomendación. Los productos se ordenan según su porcentaje de recomendación, optimizando así la experiencia de descubrimiento de productos para el usuario objetivo.

**1. Inicialización y Recuperación de Interacciones:** En esta parte, se identifica al usuario objetivo y se recuperan todas sus interacciones con los productos, asignando un peso a cada tipo de interacción (calificación, orden realizada, añadido al lista de pedidos, interacción simple). Luego, se calcula la media de estos pesos.

---

**Algorithm 6** Inicialización y Recuperación de Interacciones

---

- 1: **Entrada:** ID del usuario objetivo *user\_id*
  - 2: **Salida:** Media de los pesos de las interacciones del usuario objetivo *mean<sub>target</sub>*
  - 3: **Proceso:**
  - 4: Seleccionar el usuario objetivo *target* por su ID
  - 5: Encontrar todos los productos *P* con los que el usuario objetivo ha interactuado
  - 6: **for** cada producto *p* en *P* **do**
  - 7:     Asignar un peso *w<sub>target,p</sub>* a la interacción de *target* con *p*
  - 8:     **Si calificó:** *w<sub>target,p</sub>* = calificación
  - 9:     **Si realizó una orden:** *w<sub>target,p</sub>* = 4
  - 10:    **Si añadió al carrito:** *w<sub>target,p</sub>* = 3
  - 11:    **Si interactuó:** *w<sub>target,p</sub>* = 1
  - 12: **end for**
  - 13: Calcular la media *mean<sub>target</sub>* de los pesos *w<sub>target,p</sub>*
- 

**2. Identificación de Usuarios Similares:** En esta parte, se identifican otros usuarios que han interactuado con los mismos productos. Se asignan pesos a sus interacciones y se calcula la media de estos pesos para cada uno de ellos.

**Algorithm 7** Identificación de Usuarios Similares

---

```

1: Entrada: Productos  $P$  y media de los pesos del usuario objetivo  $mean_{target}$ 
2: Salida: Media de los pesos de las interacciones de otros usuarios  $mean_{other}$ 
3: Proceso:
4: Encontrar otros usuarios  $U$  que han interactuado con los productos en  $P$ 
5: for cada usuario  $other$  en  $U$  do
6:   for cada producto  $p$  en  $P$  do
7:     Asignar un peso  $w_{other,p}$  a la interacción de  $other$  con  $p$ 
8:     Si calificó:  $w_{other,p}$  = calificación
9:     Si realizó una orden:  $w_{other,p}$  = 4
10:    Si añadió al carrito:  $w_{other,p}$  = 3
11:    Si interactuó:  $w_{other,p}$  = 1
12:   end for
13:   Calcular la media  $mean_{other}$  de los pesos  $w_{other,p}$ 
14: end for

```

---

**3. Cálculo de la Similitud:** En esta parte, se calcula la similitud entre el usuario objetivo y cada uno de los otros usuarios utilizando la fórmula de correlación de Pearson.

**Algorithm 8** Cálculo de la Similitud

---

```

1: Entrada: Media de los pesos del usuario objetivo  $mean_{target}$  y media de los pesos de los otros usuarios  $mean_{other}$ 
2: Salida: Correlación de Pearson  $pearson\_correlation$  entre  $target$  y  $other$ 
3: Proceso:
4: for cada usuario  $other$  en  $U$  do
5:   Calcular la similitud entre  $target$  y  $other$  usando la fórmula de Pearson:
6:    $numerador = \sum (w_{target,p} - mean_{target}) \times (w_{other,p} - mean_{other})$ 
7:    $denom\_target = \sum (w_{target,p} - mean_{target})^2$ 
8:    $denom\_other = \sum (w_{other,p} - mean_{other})^2$ 
9:    $denominador = \sqrt{denom\_target \times denom\_other}$ 
10:   $pearson\_correlation = \frac{numerador}{denominador}$ 
11:  Almacenar  $pearson\_correlation$  para  $other$ 
12: end for

```

---

**4. Selección de Vecinos y Recomendación de Productos:** En esta parte, se seleccionan los usuarios más similares al usuario objetivo y se recomiendan productos basados en sus interacciones, calculando la frecuencia y el porcentaje de recomendación para cada producto.

**Algorithm 9** Selección de Vecinos y Recomendación de Productos

- 1: **Entrada:** Correlaciones de Pearson *pearson\_correlation*
- 2: **Salida:** Lista de productos recomendados con sus porcentajes de recomendación
- 3: **Proceso:**
- 4: Ordenar los usuarios *other* por *pearson\_correlation* descendente y seleccionar los más similares
- 5: Encontrar productos recomendados basados en las interacciones de los usuarios más similares que *target* no haya visto
- 6: Calcular la frecuencia de recomendación para cada producto
- 7: Normalizar la frecuencia de recomendación y calcular el porcentaje de recomendación
- 8: Ordenar los productos recomendados por porcentaje de recomendación descendente
- 9: Retornar los productos recomendados con sus porcentajes de recomendación

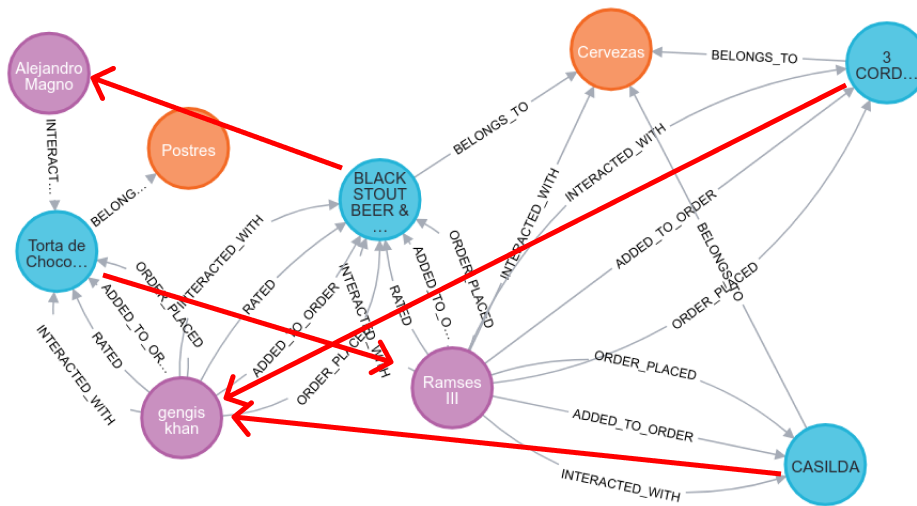
**Resultados obtenidos**

Figura 4.14: Modelo de grafo del sistema indicando productos que se recomienda al usuario

Las flechas rojas gruesas que van desde el producto a los usuarios indican que estos productos son recomendados para esos usuarios.

**Ejemplo de Recomendación:**

- “Alejandro Magno” ha interactuado con “Torta de Chocolate”.
- “Ramses III” y “Gengis Khan” han interactuado con “BLACK STOUT BCC”. Además, “Gengis Khan” ha interactuado con “Torta de Chocolate” y “Ramses III” con “CASILDA” y “3 CORDILLERA NEGRA”.

El sistema de recomendación sugiere el producto “BLACK STOUT BCC” para “Alejandro Magno”, “Torta de Chocolate” para “Ramses III”, y “CASILDA” y “3 CORDILLERA NEGRA” para “Gengis Khan”, como está indicado por las flechas rojas.

The screenshot shows the Neo4j browser interface. At the top, a Cypher query is entered: `neo4j$ MATCH (target:User {id: '578'}) MATCH (target)-[r:INTERACTED_WITH|RATED|ADDED_TO_ORDER|ORDER_PLACED]→(p:P...`. Below the query, a table displays the results:

RecommendationID	RecommendationName	Frequency	Percentage
"66"	"Torta de Chocolate"	4	100.0

Below the table, the query is expanded to show the full code:

```
1 MATCH (u:User {id: '578'})
2 RETURN u.nickname
3
```

The results section shows a table with two rows:

u.nickname
"Ramses III"

Figura 4.15: Recomendaciones de “Ramses III.”<sup>en</sup> Neo4j browser

The screenshot shows the Neo4j browser interface. At the top, a Cypher query is entered: `neo4j$ MATCH (target:User {id: '580'}) MATCH (target)-[r:INTERACTED_WITH|RATED|ADDED_TO_ORDER|ORDER_PLACED]→(p:P...`. Below the query, a table displays the results:

RecommendationID	RecommendationName	Frequency	Percentage
"70"	"BLACK STOUT BEER & COFFEE COMPANY"	4	100.0

Below the table, the query is expanded to show the full code:

```
1 MATCH (u:User {id: '580'})
2 RETURN u.nickname
3
```

The results section shows a table with two rows:

u.nickname
"Alejandro Magno"

Figura 4.16: Recomendaciones de “Alejandro Magno.”<sup>en</sup> Neo4j browser

neo4j\$ MATCH (target:User {id: '579'}) MATCH (target)-[r:INTERACTED\_WITH|RATED|ADDED\_TO\_ORDER|ORDER\_PLACED]→(p:P...

RecommendationID	RecommendationName	Frequency	Percentage
"107"	"CASILDA"	3	50.0
"101"	"3 CORDILLERA NEGRA"	3	50.0

MAX COLUMN WIDTH:

```

1 MATCH (u:User {id: '579'})
2 RETURN u.nickname
3

```

u.nickname
"gengis khan"

Figura 4.17: Recomendaciones de “gengis khan.”<sup>en</sup> Neo4j browser

La figura 4.17 muestra el resultado del algoritmo de recomendación mostrando los productos recomendado por su frecuencia y el porcentaje de los productos recomendados.

**Frecuencia:** La frecuencia se refiere al número de veces que un producto es recomendado a partir de las interacciones de los usuarios similares al usuario objetivo. Este conteo indica cuántos usuarios similares han interactuado con un producto específico, sugiriendo su relevancia para el usuario objetivo.

**Porcentaje:** El porcentaje mide la popularidad relativa de cada producto recomendado en comparación con todos los productos recomendados. Se calcula dividiendo la frecuencia del producto por la frecuencia total de todas las recomendaciones y multiplicando el resultado por 100. Esto se expresa mediante la fórmula:

$$\text{Porcentaje} = \left( \frac{\text{Frecuencia del producto}}{\text{Frecuencia total}} \right) \times 100$$

Un mayor porcentaje indica que el producto es más popular entre los usuarios similares y, por tanto, es más probable que sea de interés para el usuario objetivo.



### Algoritmo de Popularidad

Este algoritmo se implementa en el servicio de PopularService y presenta los productos populares a todos los usuarios de la aplicación, independientemente de sus preferencias individuales. El algoritmo de popularidad, que se puede considerar una recomendación no personalizada, es útil para evitar el problema del arranque en frío.

---

**Algorithm 10** Algoritmo de recomendación basado en popularidad

---

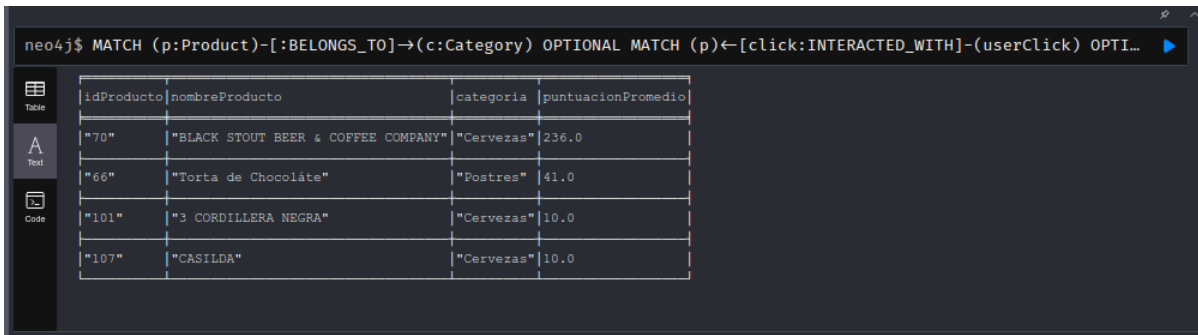
```

1: Input: Ninguno
2: Output: Lista de productos populares
3: Paso 1: Recopilación de interacciones con productos
4:  $productos \leftarrow obtenerTodosLosProductos()$ 
5: for cada producto en  $productos$  do
6:    $clicks \leftarrow contarClicks(producto)$ 
7:    $ratingSum \leftarrow sumarPuntuaciones(producto)$ 
8:    $ratingsCount \leftarrow contarPuntuaciones(producto)$ 
9:    $adds \leftarrow contarAdiciones(producto)$ 
10:   $orders \leftarrow contarOrdenes(producto)$ 
11:   $puntuacionTotal \leftarrow clicks + 3 \times ratingSum + 5 \times adds + 4 \times orders$ 
12:   $asignarPuntuacion(producto, puntuacionTotal)$ 
13: end for
14: Paso 2: Ordenar y seleccionar productos más populares
15:  $productosPopulares \leftarrow ordenarPorPuntuacion(productos)$ 
16:  $topProductos \leftarrow seleccionarTopN(productosPopulares, 10)$ 
17: return  $topProductos$ 

```

---

### Resultados obtenidos



The screenshot shows the Neo4j Browser interface. At the top, a Cypher query is entered: `neo4j$ MATCH (p:Product)-[:BELONGS_TO]->(c:Category) OPTIONAL MATCH (p)-[:click:INTERACTED_WITH]-(userClick) OPTI...`. Below the query, a table displays the results of the query. The table has four columns: `idProducto`, `nombreProducto`, `categoria`, and `puntuacionPromedio`. The results are as follows:

idProducto	nombreProducto	categoria	puntuacionPromedio
"70"	"BLACK STOUT BEER & COFFEE COMPANY"	"Cervezas"	236.0
"66"	"Torta de Chocolate"	"Postres"	41.0
"101"	"3 CORDILLERA NEGRA"	"Cervezas"	10.0
"107"	"CASILDA"	"Cervezas"	10.0

Figura 4.18: Resultados de los productos más populares del grafo anterior en Neo4j Browser

En la captura de pantalla, Figura 4.18, presenta los resultados de los productos más populares del grafo anterior, Figura 1.14. los resultados se muestran en la columna.

**puntuacionPromedio:** Esta puntuación no se calcula simplemente como un promedio de calificaciones individuales, sino que se trata de una métrica compuesta que tiene en cuenta diferentes tipos de interacciones de los usuarios con los productos.

El algoritmo utilizado para calcular la `puntuacionPromedio` se basa en las siguientes interacciones:

- *Clicks* (`clicks`): Número de veces que un usuario ha interactuado con el producto.

- *Ratings* (*ratingSum* y *ratingsCount*): Suma de las puntuaciones dadas por los usuarios y el conteo de esas calificaciones.
- *Adds* (*adds*): Número de veces que un producto ha sido añadido a una orden.
- *Orders* (*orders*): Número de veces que se ha colocado una orden que incluye el producto.

La puntuación total para cada producto se calcula con la fórmula:

$$\text{PuntuacionTotal} = \text{clicks} + 3 \times \text{ratingSum} + 5 \times \text{adds} + 4 \times \text{orders}$$

Finalmente se obtiene como resultados el promedio de estas puntuaciones totales para cada producto, redondeado a dos decimales.

Por ejemplo, en la imagen, Figura 4.18 se observa que el producto *"BLACK STOUT BEER & COFFEE COMPANY"* tiene una puntuación promedio de 236.0, lo que indica un alto nivel de interacción y valoración positiva por parte de los usuarios, en comparación con otros productos como *"Torta de Chocolate"* y *"3 CORDILLERA NEGRA"*, que tienen puntuaciones de 41.0 y 10.0 respectivamente. Esta métrica compuesta permite identificar cuáles productos son más valorados y preferidos por los usuarios dentro de la plataforma, considerando múltiples dimensiones de interacción y satisfacción del usuario.

## 4.4. Servicio de Búsqueda

La implementación del servicio de búsqueda se realizó mediante el uso de librerías de procesamiento de lenguaje natural (PLN) en Python. Una librería importante utilizada fue spaCy.[36] La incorporación de un sistema de búsqueda que facilita no solo la búsqueda de productos específicos, sino también, de sus detalles, es esencial para que el usuario se sienta seguro al buscar productos.

El uso de PLN sirvió para normalizar el texto de las consultas y los productos, así como lematizar y eliminar stopwords, lo que permite realizar búsquedas más precisas y relevantes.

### 4.4.1. Implementación

#### Paso 1: Normalización del Texto

**Descripción:** Independientemente de las variaciones en la entrada de los usuarios, se normaliza el texto removiendo diacríticos (tildes), puntuación, y convirtiendo todo el texto a minúsculas. Además, se tokeniza y lematiza el texto utilizando el modelo de spaCy.

1. *Remoción de diacríticos y puntuación:* Eliminar todos los signos de puntuación y acentos para homogeneizar el texto.
2. *Tokenización y lematización:* Utilizar spaCy para descomponer el texto en tokens y reducir cada palabra a su forma base o lema.
3. *Normalización:* Convertir todos los tokens a minúsculas y reunirlos de nuevo en una cadena de texto normalizada para la búsqueda.

#### Paso 2: Búsqueda en Productos

**Descripción:** Este paso verifica si todos los tokens de la consulta de búsqueda están presentes en el texto normalizado del producto. Esto permite una búsqueda más precisa y relevante.

**Paso 3: Filtrar Productos que Coincidan con la Consulta**

**Descripción:** Una vez que se ha realizado la normalización y tokenización, este paso filtra los productos de la base de datos que coincidan con la consulta normalizada del usuario.

**Implementación:**

1. *Normalizar la consulta de búsqueda:* Preparar la consulta del usuario aplicando los mismos pasos de normalización usados en los productos.
2. *Obtener todos los productos de la base de datos:* Recuperar todos los productos disponibles para ser buscados.
3. *Normalizar el texto de cada producto:* Aplicar el proceso de normalización a la descripción de cada producto.
4. *Verificar coincidencias:* Comprobar si la consulta normalizada está presente en el texto de los productos.
5. *Devolver la lista de productos coincidentes:* Proporcionar al usuario la lista de productos que coincidan con su búsqueda.

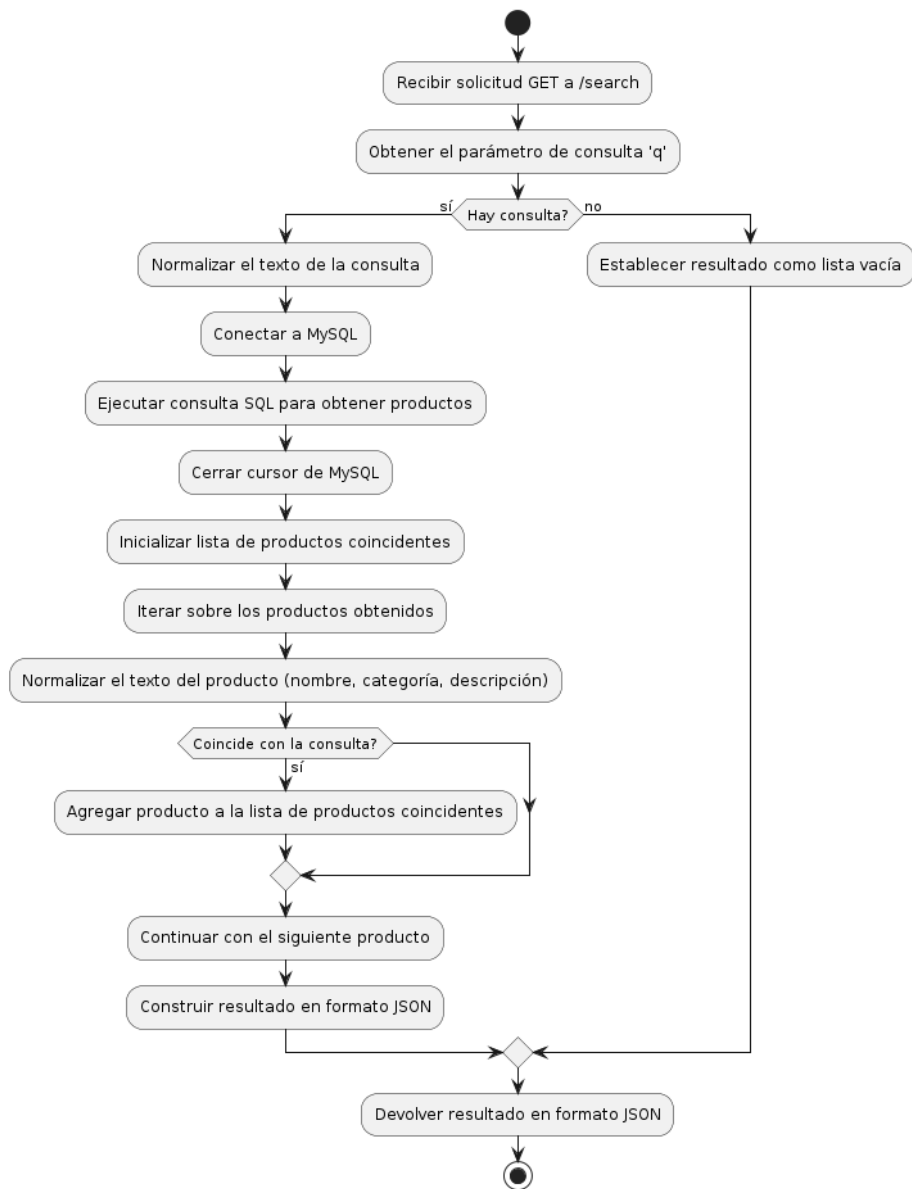
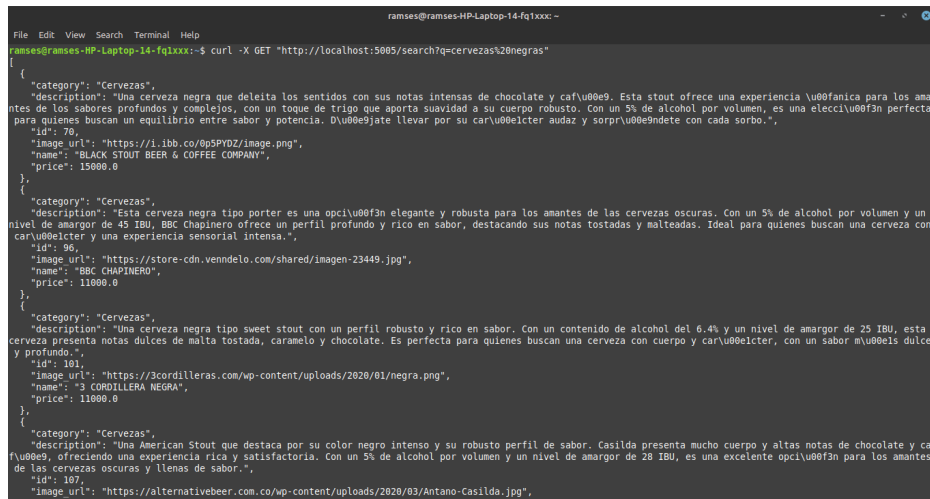


Figura 4.19: Diagrama de flujo del proceso de búsqueda en el servicio de búsqueda

## Resultados obtenidos



```

ramses@ramses-HP-Laptop-14-fq1xxx:~$ curl -X GET "http://localhost:3005/search?q=cervezas%20negras"
[{"category": "Cervezas",
  "description": "Una cerveza negra que deleita los sentidos con sus notas intensas de chocolate y caf\u00e9. Esta stout ofrece una experiencia \u00fanica para los amantes de los sabores profundos y complejos, con un toque de trigo que aporta suavidad a su cuerpo robusto. Con un 5% de alcohol por volumen, es una elecci\u00f3n perfecta para quienes buscan un equilibrio entre sabor y potencia. D\u00fabate llevar por su car\u00e1cter audaz y sorprende con cada sorbo.",
  "id": 70,
  "image url": "https://i.ibb.co/8p5PYDZ/image.png",
  "name": "BLACK STOUT BEER & COFFEE COMPANY",
  "price": 15000.0
}, {"category": "Cervezas",
  "description": "Esta cerveza negra tipo porter es una opci\u00f3n elegante y robusta para los amantes de las cervezas oscuras. Con un 5% de alcohol por volumen y un nivel de amargor de 45 IBU, BBC Chapinero ofrece un perfil profundo y rico en sabor, destacando sus notas tostadas y malteadas. Ideal para quienes buscan una cerveza con car\u00e1cter y una experiencia sensorial intensa.",
  "id": 90,
  "image url": "https://store-cdn.venndelo.com/shared/imagen-23449.jpg",
  "name": "BBC CHAPINERO",
  "price": 11000.0
}, {"category": "Cervezas",
  "description": "Una cerveza negra tipo sweet stout con un perfil robusto y rico en sabor. Con un contenido de alcohol del 6.4% y un nivel de amargor de 25 IBU, esta cerveza presenta notas dulces de malta tostada, caramelo y chocolate. Es perfecta para quienes buscan una cerveza con cuerpo y car\u00e1cter, con un sabor m\u00fas dulce y profundo.",
  "id": 101,
  "image url": "https://3cordilleras.com/wp-content/uploads/2020/01/negra.png",
  "name": "3 CORDILLERA NEGRA",
  "price": 11000.0
}, {"category": "Cervezas",
  "description": "Una American Stout que destaca por su color negro intenso y su robusto perfil de sabor. Casilda presenta mucho cuerpo y altas notas de chocolate y caf\u00e9, ofreciendo una experiencia rica y satisfactoria. Con un 5% de alcohol por volumen y un nivel de amargor de 28 IBU, es una excelente opci\u00f3n para los amantes de las cervezas oscuras y llenas de sabor.",
  "id": 107,
  "image url": "https://alternativebeer.com.co/wp-content/uploads/2020/03/Antano-Casilda.jpg"}]
```

Figura 4.20: Captura de pantalla prueba Curl a la Api de búsqueda

En la captura de pantalla, Figura 4.19 se indican los resultados GET de la API de búsqueda implementada para la consulta de “Cervezas Negras”. Los resultados muestran los productos similares que coinciden con esta búsqueda. Este resultado refleja la capacidad de la API para filtrar y presentar productos específicos basados en las características solicitadas, en este caso, productos de cervezas negras con sus respectivas características.

## 4.5. Servicio de órdenes

### 4.5.1. Implementación

Se decidió gestionar las órdenes de los clientes en OrderService, permitiendo que los productos específicos solicitados lleguen al administrador de manera instantánea mediante el uso de WebSockets [37]. Esto permite decidir si almacenarlas en una página de historial de ventas.

#### Paso 1: Configuración Inicial

Configurar la aplicación, el servidor HTTP, WebSocket y la conexión a la base de datos MySQL.

#### Paso 2: Manejo de Conexiones WebSocket

Gestionar la conexión y comunicación de WebSocket para permitir la transmisión de mensajes en tiempo real.

#### Paso 3: Manejo de la Creación de Órdenes

Gestionar la creación de órdenes, validando los campos requeridos y almacenando los datos en la base de datos MySQL.

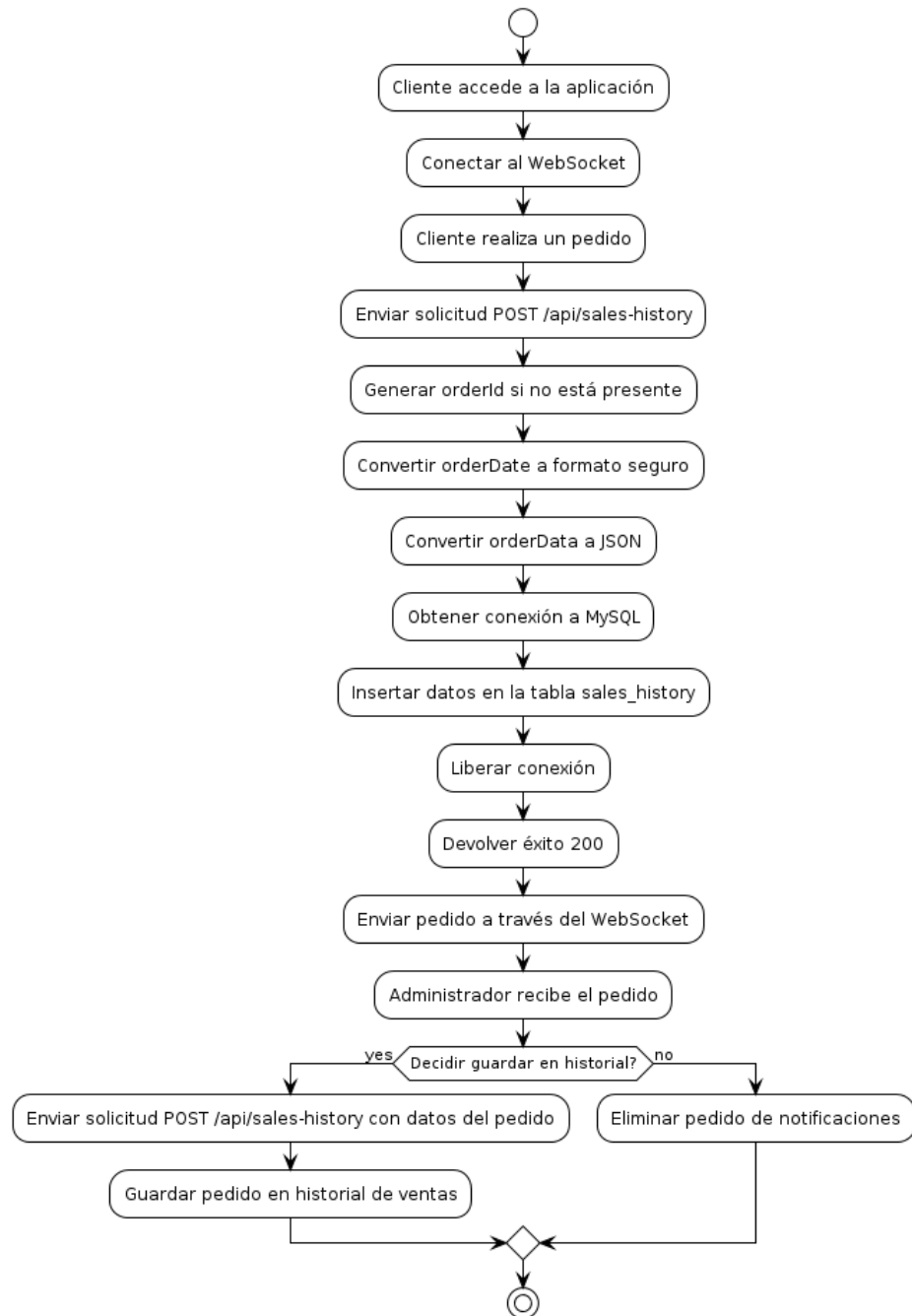


Figura 4.21: Diagrama de flujo del proceso del algoritmo de ordenes

## Capítulo 5

# Despliegue

### 5.1. Proceso de Despliegue de la Aplicación Web

El proceso de despliegue inicial de la aplicación web para Beer & Coffee Company se llevó a cabo utilizando tecnologías como Docker, Nginx y Ngrok. Este despliegue inicial facilitó las pruebas de aceptación y usabilidad, permitiendo recibir retroalimentación. A continuación, se describe detalladamente cada etapa del despliegue.

#### 5.1.1. Configuración de Docker

Se utilizó Docker para crear contenedores que permiten un entorno aislado y consistente para cada microservicio, lo que facilita la gestión y escalabilidad del sistema. Docker Compose fue empleado para orquestar todos los servicios involucrados, tales como el servicio de autenticación, gestión de productos, frontend, servicio de órdenes, servicio de búsqueda, servicio de popularidad, servicio de recomendaciones e interacciones, así como las bases de datos MySQL y Neo4j.[38]

Cada microservicio fue configurado con sus propias variables de entorno y puertos, asegurando que todos los servicios se comuniquen a través de una red Docker compartida. Esta arquitectura modular permitió desplegar y escalar cada componente de manera independiente, optimizando así el uso de recursos y mejorando la resiliencia del sistema.

#### 5.1.2. Configuración de Nginx

Nginx se utilizó como servidor proxy inverso y balanceador de carga. Esta configuración ayudó a distribuir el tráfico entre los diferentes microservicios, mejorando la eficiencia y la seguridad del sistema. Nginx gestionó las solicitudes HTTP entrantes y las redirigió al microservicio correspondiente, basado en las reglas definidas en su configuración. Además, Nginx se encargó de gestionar la compresión y el almacenamiento en caché, optimizando el rendimiento y reduciendo la carga en los servicios backend.[39]

#### 5.1.3. Integración con Ngrok

Ngrok se utilizó para exponer los servicios locales a Internet, facilitando las pruebas y el acceso remoto durante la fase de desarrollo. Al configurar Ngrok, se creó un túnel que permitió acceder a la aplicación web desde cualquier lugar.

Ngrok generó una URL pública temporal que redirigía el tráfico hacia el servidor Nginx en el entorno

local, lo cual fue particularmente útil para recibir retroalimentación en tiempo real de los usuarios y para realizar demostraciones a los clientes.[40]

#### 5.1.4. Flujo de Despliegue

- **Construcción de Imágenes Docker:** Se construyeron las imágenes Docker para cada microservicio utilizando los Dockerfiles correspondientes. Esto incluyó la instalación de todas las dependencias necesarias y la configuración del entorno de ejecución.
- **Orquestación con Docker Compose:** Se utilizó Docker Compose para definir y gestionar el conjunto de servicios, asegurando que todos los contenedores se inicien en el orden correcto y con las dependencias adecuadas.
- **Configuración de Nginx:** Nginx se configuró para actuar como un proxy inverso, distribuyendo las solicitudes entrantes a los microservicios correspondientes y gestionando la compresión y el almacenamiento en caché.
- **Exposición a Internet con Ngrok:** Se configuró Ngrok para crear un túnel seguro hacia el servidor Nginx local, permitiendo el acceso remoto y facilitando las pruebas y demostraciones.

Se anticipa que, en el futuro, el despliegue completo se realice utilizando servicios en la nube como AWS, Google Cloud o Azure para mantener el servicio en línea.



## Capítulo 6

# Pruebas

### 6.1. Pruebas de Usabilidad y Aceptación

#### 6.1.1. Introducción

El objetivo de las pruebas de usabilidad y aceptación fue evaluar la precisión, funcionalidad, utilidad y satisfacción general de los usuarios con respecto a la aplicación web de BCC. Las pruebas se realizaron mediante encuestas dirigidas a los clientes del establecimiento, recopilando sus opiniones y experiencias con diferentes aspectos de la aplicación.

#### 6.1.2. Metodología

Se diseñó un cuestionario en Google Forms con preguntas específicas sobre la usabilidad y la aceptación de diversas funcionalidades de la aplicación web desde la perspectiva del cliente. La encuesta fue distribuida a un grupo representativo de usuarios, obteniendo un total de 30 respuestas. Antes de responder a la encuesta, a los usuarios se les proporcionó un enlace a la aplicación para que pudieran navegar por ella. Además, se les entregó un manual de usuario detallado para guiar su experiencia y asegurar que pudieran explorar todas las funcionalidades de la aplicación.

#### 6.1.3. Pruebas de Usabilidad

##### Precisión de los Detalles del Producto

**Pregunta:** “¿Qué tan preciso te parece cómo se muestran los detalles del producto en la base de datos?”

**Resultados:** La mayoría de los participantes consideran que los detalles del producto son normales y precisos en primer lugar, preciso, con una minoría que los encuentra impreciso y muy impreciso. Esto indica un estado de confianza moderado con muchas condiciones para mejorar.

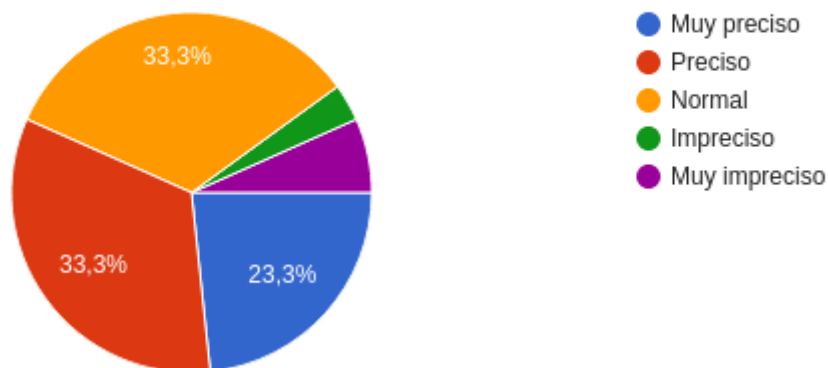


Figura 6.1: Gráfica de la precisión de los detalles del producto

### Comentarios sobre la Interfaz

Las respuestas abiertas y los comentarios adicionales sugieren que hay una preocupación por la intuitividad y facilidad de uso de la interfaz.

#### 6.1.4. Pruebas de Aceptación

##### Utilidad de las Recomendaciones

**Pregunta:** “¿Qué tan útiles encuentran los usuarios las recomendaciones para descubrir nuevos productos que les interesan?”

**Resultados:** La mayoría de los usuarios encuentra las recomendaciones útiles o muy útiles, lo que indica que el sistema de recomendaciones está cumpliendo su propósito.

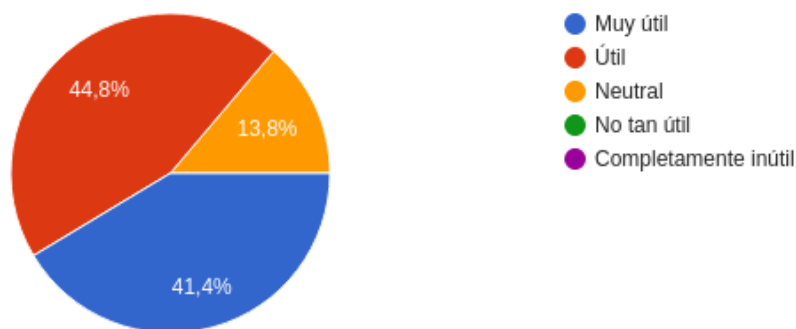


Figura 6.2: Gráfica de la utilidad de las recomendaciones

### Satisfacción con las Recomendaciones

**Pregunta:** “¿Cuál es el nivel de satisfacción general de los usuarios con las recomendaciones proporcionadas?”

**Resultados:** La mayoría de los participantes están satisfechos con las recomendaciones, mostrando con esta funcionalidad.

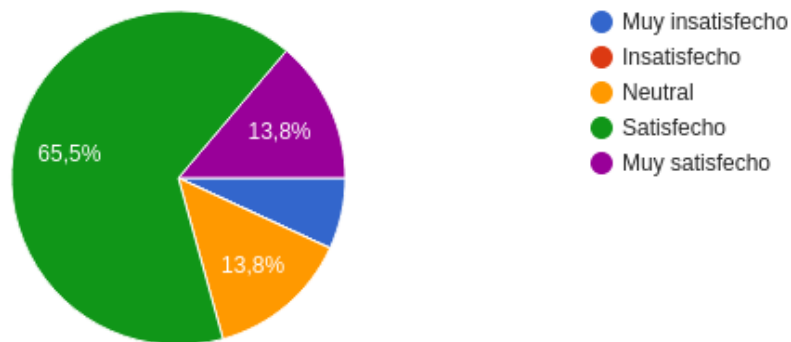


Figura 6.3: Gráfica de la satisfacción con las recomendaciones

### Influencia de las Recomendaciones

**Pregunta:** “¿En qué medida influyen las recomendaciones en las decisiones de compra o interacción de los usuarios con los productos?”

**Resultados:** La mayoría de los usuarios indica que las recomendaciones influyen en sus decisiones de compra o interacción, evidenciando la efectividad de las recomendaciones pero con una neutralidad que llama la atención indicando que el sistema de recomendaciones aun falta mucho por mejorar.

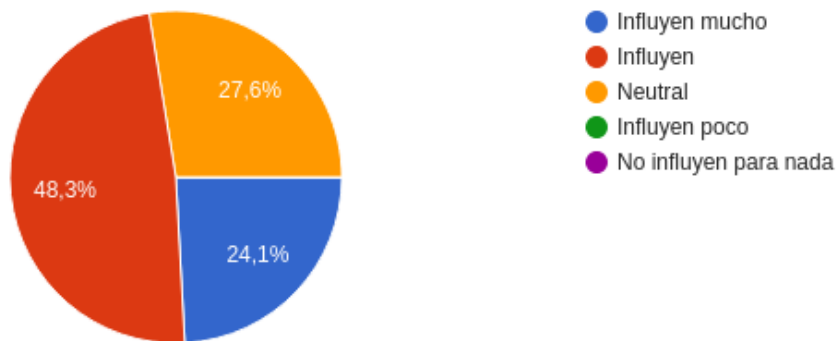


Figura 6.4: Gráfica de la influencia de las recomendaciones

### 6.1.5. Análisis de los Resultados Más Bajos

Se analizaron las respuestas que indicaban percepciones medias o bajas para identificar áreas de mejora potencial en la aplicación. Para la parte de pruebas la aplicación muestra una precisión neutral en la visualización de detalles de productos y un sistema de recomendaciones efectivo para los usuarios.

A pesar de la satisfacción general, hay áreas que pueden mejorarse, especialmente en la precisión de los detalles del producto y la utilidad de las recomendaciones.

Se recomienda continuar monitoreando y optimizando las funcionalidades clave, basándose en el feedback de los usuarios para mantener y mejorar la experiencia del usuario. Específicamente, se deben considerar los comentarios sobre la interfaz para hacerla más intuitiva y fácil de usar.

## Capítulo 7

# Conclusiones

Este proyecto aborda la creación de una aplicación web para BCC que permita a los clientes sentirse seguros al elegir productos, gracias a sistemas basados en recomendaciones y una interfaz que permite una visualización más detallada de los productos. Además, proporciona a los clientes un motor de búsqueda donde pueden encontrar productos que se ajusten a sus gustos específicos, brindándoles resultados satisfactorios. El acceso al catálogo de productos de BCC desde dispositivos móviles elimina la necesidad de una carta física. Esta transición no solo mejora la comodidad del usuario, sino que también permite una exploración más detallada y visualmente atractiva de los productos que se venden en el establecimiento. Los clientes pueden revisar descripciones, ver imágenes y recibir recomendaciones que les ayuden a tomar decisiones informadas y seguras.

En cuanto al sistema de recomendaciones basado en filtrado colaborativo, es importante señalar que, para los nuevos usuarios, este sistema inicialmente no puede predecir sus gustos debido a la falta de historial de interacciones. Para estos usuarios, se utilizan recomendaciones no personalizadas, que les permiten descubrir productos populares basados en las preferencias generales de otros clientes. Por otro lado, para los usuarios recurrentes, el sistema emplea algoritmos de filtrado colaborativo que ofrecen sugerencias alineadas con sus gustos y preferencias previas. Aunque las interacciones actuales del cliente, como clics en productos, calificaciones, agregar a la lista de pedidos y realizar pedidos, son básicas, han demostrado ser efectivas para generar recomendaciones relevantes. No obstante, el sistema tiene la capacidad de expandirse para considerar muchas más variables, y no se limita solo a operaciones matemáticas entre grafos, sino que también puede beneficiarse de la optimización de los algoritmos. Aunque existen múltiples tipos de algoritmos que podrían ofrecer mejores o peores resultados, según las pruebas, la elección actual ha demostrado ser adecuada y proporciona resultados satisfactorios.

Además, el sistema de búsqueda de la aplicación utiliza procesamiento de lenguaje natural (PLN) para permitir búsquedas efectivas no solo por el nombre del producto, sino también, por sus características. Esta funcionalidad facilita a los clientes encontrar productos específicos según la composición del producto, mejorando la búsqueda de acuerdo a los gustos del cliente.

La elección de una arquitectura de servicios facilita la escalabilidad del sistema, permitiendo enfocar en un solo servicio a la vez, sin comprometer el sistema completo. Con base en esto, se permite la integración de nuevas funcionalidades de manera modular. Esta flexibilidad tecnológica prepara el camino para futuras expansiones y mejoras continuas en el servicio.

El despliegue inicial de la aplicación, utilizando tecnologías como Docker, Nginx y Ngrok, ha permiti-

do realizar pruebas y obtener retroalimentación valiosa de los usuarios. Esta retroalimentación ha sido crucial para identificar áreas de mejora y asegurar que la aplicación cumpla con las expectativas de los clientes.

La implementación de algoritmos de filtrado colaborativo basados en la similitud de Pearson en la aplicación web de BCC ha producido resultados cuantificados mediante métricas de **frecuencia** y **porcentaje**. En la Figura 4.16 se presentan los resultados de este algoritmo de recomendación, mostrando la frecuencia y el porcentaje de los productos recomendados a los usuarios. Adicionalmente, en la Figura 4.18 se ilustran los resultados del algoritmo de popularidad, los cuales se calculan como el promedio ponderado de las interacciones de los usuarios con los productos.

El desarrollo de este proyecto me ha dejado importantes enseñanzas y experiencias que han enriquecido mi formación profesional. Aprendí a sacar lo mejor de cada herramienta y lenguaje de programación, adaptándome a las particularidades de cada uno para maximizar su eficiencia. Este proceso me permitió profundizar en arquitecturas de software y en cómo se comunica la información detrás de una aplicación web, lo que resultó esencial para la correcta implementación y despliegue del sistema. Asimismo, el manejo de variables de entorno en imágenes de Docker y la comunicación efectiva con mi director fueron clave para el éxito del proyecto. Este trabajo también me enseñó la importancia de ser ordenado y de enfocarme en una sola tarea específica a la vez, lo que mejoró significativamente mi productividad. A través de las investigaciones y la selección de teorías necesarias, viví una experiencia enriquecedora, comprendiendo que la tecnología, bien aplicada, puede mejorar la calidad de vida de los clientes. Además, las pruebas realizadas me enseñaron que ningún software es perfecto y siempre hay margen para mejoras continuas.

En definitiva, la aplicación web desarrollada para BCC transformaría la experiencia del cliente al ofrecer una tecnología accesible y efectiva para la selección de pedidos. El sistema de recomendaciones, en particular, ayudaría a los clientes a navegar por la amplia variedad de productos y tomar decisiones informadas y satisfactorias. Aunque hay áreas que requieren desarrollo y optimización adicionales, los cimientos establecidos prometen una experiencia de usuario cada vez mejor y también sientan las bases para futuras innovaciones y mejoras continuas en el servicio ofrecido por parte de Beer & Coffee Company a sus clientes.

## Capítulo 8

# Trabajos futuros

La aplicación web de BCC ha sido diseñada con una arquitectura de servicios, lo que permite una expansión para cualquier otra funcionalidad que se necesite en el futuro. Esta flexibilidad tecnológica prepara el camino para integraciones adicionales sin afectar negativamente el sistema existente. A continuación, se detallan los trabajos futuros.

### 8.0.1. Mejoramiento del Sistema de Recomendaciones

El sistema de recomendaciones puede beneficiarse significativamente de una serie de mejoras:

- **Investigación y Implementación de Machine Learning Avanzado:** Explorar y aplicar técnicas avanzadas de machine learning, incluyendo redes neuronales, para mejorar la precisión y relevancia de las recomendaciones personalizadas[41].
- **Incorporación de Nuevas Variables:** Añadir variables adicionales que alimenten el sistema de recomendación, como los comentarios de los clientes sobre los productos, y factores externos como el clima[5]. Por ejemplo, en días fríos, el sistema podría priorizar la recomendación de café caliente, mientras que en días calurosos, podría sugerir cerveza artesanal fría.
- **Análisis de Sentimiento:** Implementar análisis de sentimiento en los comentarios de los productos para ajustar las recomendaciones basadas en las percepciones positivas o negativas de otros usuarios.

### 8.0.2. Optimización del Sistema de Búsqueda

Mejorar el sistema de búsqueda es esencial para proporcionar una experiencia de usuario más fluida y efectiva:

- **Predicción de Texto Automática:** Integrar funciones de predicción de texto automático que sugieran términos de búsqueda a medida que el usuario escribe, facilitando una búsqueda más rápida y precisa.
- **Pruebas de Rendimiento:** Realizar pruebas exhaustivas para evaluar y optimizar la fluidez de los microservicios, asegurando que todas las consultas se procesen rápidamente y sin errores.

### 8.0.3. Migración a la Nube

Para mejorar la escalabilidad, la disponibilidad y la seguridad del sistema, se recomienda la migración a una infraestructura en la nube:

- **Plataformas en la Nube:** Considerar la utilización de servicios en la nube como AWS, Google Cloud o Azure para alojar los microservicios, lo que también facilitará la gestión y el monitoreo continuo del sistema.

#### 8.0.4. Mejora de la Interfaz de Usuario

La interfaz de usuario juega un papel crucial en la experiencia del cliente y debe ser optimizada continuamente:

- **Retroalimentación del Usuario:** Basar las mejoras de la interfaz en la retroalimentación recibida de los usuarios. Esto incluye ajustar el diseño para hacerlo más intuitivo y visualmente atractivo, mejorando así la navegación y la interacción con la aplicación.
- **Pruebas de Usabilidad Continuas:** Implementar un ciclo de pruebas de usabilidad constante para identificar y corregir problemas rápidamente, asegurando que la interfaz evolucione junto con las necesidades de los usuarios.

#### 8.0.5. Integración de Pagos en Línea

Desarrollar e implementar un sistema de pagos en línea seguro permitirá a los usuarios realizar transacciones directamente desde la aplicación, mejorando la comodidad y la experiencia de compra.

#### 8.0.6. Sistema de Registro y Login Más Completo

Implementar un sistema de registro y login más completo permitirá recopilar información detallada sobre los clientes. Esto no solo mejorará la seguridad y la personalización de la experiencia del usuario, sino que también proporcionará datos valiosos para optimizar las recomendaciones y otros servicios.

#### 8.0.7. Mejorar el Panel de Administradores

Desarrollar un panel de administración más completo permitirá a BCC gestionar de manera eficiente el inventario y obtener análisis detallados de ventas. Un dashboard con métricas clave ayudará a los administradores a tomar decisiones informadas y estratégicas.

#### 8.0.8. Evaluación y Optimización Continuas

Mantenerse actualizado con las nuevas tecnologías y metodologías en el campo de los sistemas de recomendación y la inteligencia artificial, evaluando su aplicabilidad e integrándolas cuando sea pertinente, es crucial para mantener la relevancia y efectividad de la aplicación.



# Referencias

- [1] Beer & Coffee Company. *Beer & Coffee Company: Instagram Profile*. 2024. URL: <https://www.instagram.com/bcctulua/?hl=es>.
- [2] B. Schwartz. *The Paradox of Choice: Why More Is Less*. New York: Harper Perennial, 2004.
- [3] J. Smith y R. Doe. “The Effect of Choice Overload on Consumer Satisfaction”. En: *Journal of Consumer Research* 21.4 (dic. de 2019), págs. 600-608.
- [4] BBVA. *¿Qué es la metodología ‘agile’? ¿Revolución de las formas de trabajo?* 2024. URL: <https://www.bbva.com/es/innovacion/metodologia-agile-la-revolucion-las-formas-trabajo/>.
- [5] Francesco Ricci, Lior Rokach y Bracha Shapira, eds. *Recommender Systems Handbook*. Springer, 2022.
- [6] Neo4j Developer Blog. *Exploring Practical Recommendation Systems in Neo4j*. 2024. URL: <https://neo4j.com/developer-blog/exploring-practical-recommendation-systems-in-neo4j/>.
- [7] Neo4j Use Cases. *Real-Time Recommendation Engine*. 2024. URL: <https://neo4j.com/blog/real-time-recommendation-engine-data-science/>.
- [8] Amazon Web Services. *¿Qué es una aplicación web?* 2024. URL: <https://aws.amazon.com/es/what-is/web-application/>.
- [9] Martin Fowler. *Software Architecture*. 2024. URL: <https://martinfowler.com/architecture/>.
- [10] Kim Falk. *Practical Recommender Systems*. Manning Publications, 2019.
- [11] Francesco Ricci, Lior Rokach y Bracha Shapira. *Recommender Systems Handbook*. Springer, 2015.
- [12] Paul Resnick et al. “GroupLens: An Open Architecture for Collaborative Filtering of Netnews”. En: *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. 1994, págs. 175-186.
- [13] Gerard Salton, A Wong y C.S. Yang. “A vector space model for automatic indexing”. En: *Communications of the ACM* 18.11 (1975), págs. 613-620.
- [14] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [15] Paul Jaccard. “Étude comparative de la distribution florale dans une portion des Alpes et des Jura”. En: *Bulletin de la Société Vaudoise des Sciences Naturelles* 37 (1901), págs. 547-579.
- [16] Pang-Ning Tan, Michael Steinbach y Vipin Kumar. *Introduction to Data Mining*. Pearson, 2006.
- [17] Gediminas Adomavicius y Alexander Tuzhilin. “Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”. En: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), págs. 734-749.

- [18] John S Breese, David Heckerman y Carl Kadie. “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”. En: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI '98)*. 1998, págs. 43-52.
- [19] Jerome L. Myers, Arnold D. Well y Robert F. Lorch. *Research Design and Statistical Analysis*. Routledge, 2010.
- [20] Charles Spearman. “The Proof and Measurement of Association between Two Things”. En: *The American Journal of Psychology* 15.1 (1904), págs. 72-101.
- [21] Solomon Kullback y Richard A. Leibler. “On Information and Sufficiency”. En: *The Annals of Mathematical Statistics* 22.1 (1951), págs. 79-86.
- [22] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [23] Richard W. Hamming. “Error Detecting and Error Correcting Codes”. En: *The Bell System Technical Journal* 29.2 (1950), págs. 147-160.
- [24] David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
- [25] World Wide Web Consortium. *Web Services Glossary*. 2004. URL: <https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/#webservice>.
- [26] Wikipedia contributors. *Aplicación monolítica — Wikipedia, La enciclopedia libre*. 2024. URL: [https://es.wikipedia.org/wiki/Aplicaci%C3%B3n\\_monol%C3%ADtica](https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_monol%C3%ADtica).
- [27] Martin Fowler. *Microservices*. 2024. URL: <https://martinfowler.com/microservices/>.
- [28] FreeCodeCamp. *UX vs UI: What's the Difference? Definition and Meaning*. 2024. URL: <https://www.freecodecamp.org/news/ux-vs-ui-whats-the-difference-definition-and-meaning/>.
- [29] Figma. *Figma Help Center*. Último acceso: 10 de junio de 2024. 2024. URL: <https://help.figma.com/hc/en-us>.
- [30] React. *React Documentation: Learn React*. 2024. URL: <https://react.dev/learn>.
- [31] Techopedia. *Recomendación de arranque en frío*. Último acceso: 10 de junio de 2024. 2024. URL: <https://www.techopedia.com/es/recomendacion-arranque-frio>.
- [32] Neo4j. *Cypher Query Language: Introduction and Overview*. Último acceso: 10 de junio de 2024. 2024. URL: <https://neo4j.com/docs/cypher-manual/current/introduction/cypher-overview/>.
- [33] Neo4j. *This Week in Neo4j: Time-based Graph Versioning, Pearson Coefficient, Neo4j Multi-DC*. 2023. URL: <https://neo4j.com/blog/this-week-in-neo4j-time-based-graph-versioning-pearson-coefficient-neo4j-multi-dc/>.
- [34] Neo4j Community. *Pearson Similarity*. 2023. URL: <https://community.neo4j.com/t/pearson-similarity/66337>.
- [35] Statistics How To. *Correlation Coefficient Formula*. 2024. URL: <https://www.statisticshowto.com/probability-and-statistics/correlation-coefficient-formula/>.
- [36] spaCy. *spaCy API Documentation*. 2024. URL: <https://spacy.io/api/doc>.
- [37] Github Contributors. *ws: a Node.js WebSocket library*. 2024. URL: <https://github.com/websockets/ws>.
- [38] Docker. *Docker Documentation*. 2024. URL: <https://docs.docker.com/>.
- [39] Nginx. *Nginx Documentation*. 2024. URL: <https://nginx.org/en/docs/>.

- [40] ngrok. *ngrok Documentation*. 2024. URL: <https://ngrok.com/docs/>.
- [41] TensorFlow. *Recursos para aprender ML teórico y avanzado*. Último acceso: 10 de junio de 2024. 2024. URL: <https://www.tensorflow.org/resources/learn-ml/theoretical-and-advanced-machine-learning?hl=es-419>.

## Capítulo 9

# Anexos

El código fuente del proyecto puede encontrarse en el siguiente repositorio de GitHub:

<https://github.com/happodaikarin/ProyectoFinal>

Documentación completa de los diagramas generales de la arquitectura y de los servicios específicos

<https://github.com/happodaikarin/ProyectoFinal/tree/main/documentacion>