



**INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**



Trabajo Terminal:

**“API para el Desarrollo de Sistemas de Recomendación”
2015-A056**

Presentan:

**López Garduño Blanca Azucena
Castro Esparza José Antonio
Bautista de Jesús Héctor Gerardo**

Palabras Clave: Inteligencia Artificial, Sistemas de Recomendación, Ingeniería de Software, Machine Learning

Directores:

M. en C. Rocío Reséndiz Muñoz, M. en E. Carlos Silva Sánchez

Índice general

1. Introducción	5
2. Definición del proyecto	6
2.1. Planteamiento del problema	6
2.2. Propuesta de solución	6
2.3. Objetivo general	7
2.4. Justificación	7
3. Estado del arte	8
3.1. Spotify	8
3.2. Trabajo Terminal Sistema Generador de Recomendaciones para una Tienda En-Línea de videojuegos.	8
3.3. PredictionIO	8
3.4. Easyrec	9
3.4.1. REST API	9
3.5. LensKit	9
4. Marco teórico	10
4.1. Sistemas de recomendación	10
4.1.1. Técnicas de recomendación	10
4.1.2. Comparando técnicas de recomendación	11
4.1.3. Sistemas híbridos	12
4.2. Algoritmos de recomendación	16
4.2.1. Vecinos próximos (k-NN)	16
Basado en usuario	16
Basado en item	16
4.2.2. Slope One	17
4.2.3. Factorización de matrices	17
4.2.4. Basados en contenido	19
4.2.5. Algoritmos no personalizados	19
4.3. Bases de datos orientadas a grafos	19
4.3.1. Características	20
Ventajas	20
Desventajas	20
4.4. Application Programming Interface (API)	21

5. Análisis general del proyecto	22
5.1. Características	22
5.2. Restricciones	22
5.3. Metodología	23
5.3.1. Descripción	23
5.3.2. Prototipos esperados	23
5.4. Descripción y módulos del sistema	23
5.4.1. Arquitectura general	23
5.4.2. Módulos de la API	25
Módulo de obtención de datos	25
Módulo de operaciones lógicas	25
Módulo de presentación	26
5.4.3. Aplicación final	26
Definición	26
5.5. Análisis y gestión de riesgos	26
5.5.1. Definición y clasificación	26
5.6. Tecnologías	28
6. Prototipo 1: Obtener y abstraer los datos a utilizar	38
6.1. Análisis	38
6.1.1. Objetivo	38
6.1.2. Características	38
6.1.3. Restricciones	38
6.2. Diseño	38
6.2.1. Modelo de datos	38
6.3. Resultados	39
7. Prototipo 2: Visualizar los datos en recomendaciones por contenido	43
7.1. Análisis	43
7.1.1. Objetivo	43
7.1.2. Características	43
7.1.3. Restricciones	43
7.2. Diseño	43
7.2.1. Diagrama de clases	44
7.2.2. Casos de uso	44
7.3. Resultados	46
8. Anexos	49
8.1. Categorías propuestas para el caso de estudio	49
8.2. Diccionario de Datos	51
Glosario	53
Bibliografía	56

Índice de cuadros

4.1. Comparación de las distintas técnicas de recomendación	14
4.2. Sistemas híbridos	15
5.1. Cuadro de prototipos esperados	23
5.2. Analisis de Riesgos	27
5.3. Tipos de bases de datos	29
5.4. Cuadro comparativo de gestores de bases de datos orientadas a grafos . . .	30
5.5. Cuadro de tecnologías para diseño Web	31
5.6. software para la gestión y construcción de proyectos	32
5.7. Software para el control de versiones	33
5.8. Lenguaje de desarrollo y tecnologías relacionadas	34
5.9. Tecnologías usadas	37
8.1. Categorías propuestas para el caso de estudio	50
8.2. Diccionario de datos	52

Índice de figuras

4.1. Descomposición de la matrix en el producto de otras tres	18
5.1. Diagrama general del sistema	24
6.1. Modelo general necesario para el manejo de la información	39
6.2. Modelo de datos propuesto para el caso de estudio	41
6.3. Modelo de datos implementado para el caso de estudio	42
7.1. Diagrama de clases del prototipo 2	44
7.2. Casos de uso del prototipo 2	44
7.3. Capturas de pantalla del sistema	46
7.4. Modelo de datos con funcionalidad limitada para el prototipo 2	47
7.5. Recomendación obtenida para la búsqueda de ensaladas	48

Capítulo 1

Introducción

Debido al constante crecimiento de la información que se encuentra disponible en la Internet, en los últimos años el uso de sistemas de recomendación se ha incrementado considerablemente, debido a sus características es utilizado en un sin fin de aspectos, ayudando al comercio electrónico así como en búsquedas de información al mostrar artículos relacionados que pueden ser de interés al usuario final. Esto se logra a través de la aplicación de la Inteligencia Artificial (IA) la cual por medio de ciencias como las ciencias de la computación y la lógica, estudia la creación de entidades que son capaces de resolver problemas por sí mismas utilizando el paradigma de la inteligencia humana. Existen sistemas que cuentan con herramientas de inteligencia artificial, dichos sistemas pueden ejecutar distintos procesos análogos al comportamiento humano, determinando la devolución de una respuesta por cada entrada mediante una lógica formal. [1]

El uso de la IA se ve implicado en los sistemas de recomendación, los cuales son sistemas inteligentes que proporcionan al usuario una serie de sugerencias determinadas por las características de los objetos categorizados y evaluados en el sistema, así como diferentes reglas pertenecientes a la lógica de los algoritmos de selección. Esto resulta en un conjunto de objetos que deberán adaptarse de acuerdo a los cambios en la información en el transcurso del tiempo. Cabe mencionar que los sistemas de recomendación hacen uso de conocimientos relacionados al aprendizaje máquina la cual es una disciplina que trata de que los sistemas aprendan automáticamente, es decir que debe identificar millones de datos en patrones bastante complejos por medio de un algoritmo el cual se encarga de revisar los datos y así es capaz de predecir comportamientos futuros. [2]

Debido al problema que conlleva para los programadores obtener los conocimientos necesarios al implementar un sistema de recomendación al mismo tiempo que se intenta resolver un problema en particular, en este Trabajo Terminal se propone desarrollar un conjunto de funciones, reglas, especificaciones y procedimientos (es decir, un API, por sus siglas en inglés (Application Programming Interface), que brinde dichas funciones con el fin de reducir la curva de aprendizaje del programador al momento de desarrollar sistemas de recomendación. [3] Así, al tiempo que se crean las herramientas necesarias para obtener recomendaciones de artículos, se creará un sistema de recomendación de platillos que verifique la funcionalidad desarrollada en el Trabajo Terminal.

Capítulo 2

Definición del proyecto

2.1. Planteamiento del problema

Actualmente, la cantidad de información que existe en Internet es inmensa, esto nos muestra el creciente problema para los usuarios al buscar algo que les sea relevante; entre mayor es la cantidad de información, más difícil se vuelve encontrar un artículo en específico. Es aquí donde los sistemas de recomendación tienen su razón de ser. Tan solo ejemplos claros como Google o Facebook hacen uso de este tipo de sistemas para mostrar los resultados de las búsquedas, o del contenido que se muestra en el ifeed de noticias. El uso de los sistemas de recomendación ha permitido el auge de sistemas como los son Amazon, Spotify y Netflix. En general, todo el sistema de comercio electrónico se ha visto beneficiado con el uso de los sistemas de recomendación.

Es por esto, que cada vez más sistemas utilizan este tipo de herramientas para brindar servicios añadidos hacia sus usuarios que los destaquen entre otras plataformas. Sin embargo, la construcción de un sistema de recomendación no es una tarea sencilla. Requiere de un proceso consciente de análisis para la manera en que los artículos y usuarios se verán relacionados. Así, supone tener un conocimiento de las características propias de un sistema de recomendación, los algoritmos que resuelvan la obtención de recomendaciones y al mismo tiempo, resolver el problema en particular para los artículos y usuarios, lo cual recae en una marcada curva de aprendizaje así como en un mayor tiempo de desarrollo para este tipo de sistemas.

2.2. Propuesta de solución

Como respuesta a la necesidad creciente del uso de sistemas de recomendación en diferentes ámbitos y plataformas, se propone la creación de una API, definida por un conjunto de funciones, reglas, especificaciones y procedimientos que coadyuven a la creación de sistemas de recomendación para diversos tipos de artículos, con un modelo propuesto por el equipo de trabajo, y la implementación de algoritmos de recomendación para diversos propósitos, como lo son, el filtrado por contenido y el filtrado colaborativo. Así como resolver una problemática para la recomendación de platillos en un área específica de la Ciudad de México que vería validadas las funciones de la API en cuestión.

2.3. Objetivo general

Diseñar e implementar una API que brinde funciones de abstracción y operación de datos, clasificación y análisis de los mismos a través de las características de un conjunto de datos para obtener, por medio de evaluaciones y predicciones, conjuntos de objetos que representen una recomendación.

2.4. Justificación

El uso de sistemas de recomendación se ha extendido popularmente en los últimos años, teniendo ejemplos en casi cualquier parte de la web, sobretodo su uso en sistemas de comercio electrónico, siendo menos comunes las aplicaciones en otras áreas. Sin embargo, el desarrollo de aplicaciones o sistemas que contengan dentro de sus características sistemas de tipo recomendación requiere un conocimiento especializado en este rubro por parte de los desarrolladores para construir el sistema desde cero, siendo al final componentes comerciales los que terminan formando parte del sistema completo, perdiendo así flexibilidad en la implementación y adecuándose el desarrollador a los requerimientos de los componentes comerciales para recomendaciones. Esto implica que al momento de desarrollar un sistema que resuelva una necesidad a través de un sistema de recomendaciones, el programador debe adquirir los conocimientos necesarios para desarrollar el sistema de recomendaciones al mismo tiempo que intenta resolver el problema de dominio al que se está enfrentando. La API a desarrollar permitirá al programador obtener las herramientas necesarias para la implementación de un sistema de recomendación utilizando conocimientos de la estadística para implementar sistemas de recomendación basados en contenido, colaborativos o híbridos dependiendo del contexto en que se desea utilizar a través de un modelo de datos adecuado que sea capaz de realizar dichos procedimientos y así reducir la curva de aprendizaje obligatoria durante la incursión en un dominio de aplicación no conocido para que este se centre en el problema que desee resolver y no en las herramientas necesarias para tal fin.

Dicha API brindará funcionalidades propias de los sistemas de recomendación mediante llamadas a bibliotecas que ofrecerán el acceso a servicios desde los procesos y representará un método para conseguir abstracción en la programación. El desarrollo de una API que permita proveer la infraestructura mínima necesaria para crear un sistema de recomendación funge como proyecto integrador de los conocimientos adquiridos durante el estudio de la carrera de Ingeniería en Sistemas Computacionales, para el desarrollo del citado proyecto se requiere hacer uso de los saberes adquiridos por los participantes durante su trayectoria escolar tales como los conocimientos en materia de inteligencia artificial, ingeniería de software, matemáticas discretas, matemáticas avanzadas, reconocimiento de patrones, programación orientada a objetos, entre otras. Al final el uso conjunto de los conocimientos mencionados terminará en dicha API cuyos beneficios pueden ser vistos de manera inmediata al término de su desarrollo con su verificación y validación en un caso de estudio particular.

Capítulo 3

Estado del arte

Debido al auge de los sistemas de recomendación, es posible mostrar una larga lista de servicios que utilizan este tipo de herramientas, a continuación se encuentran algunas muy relevantes. Cabe destacar que muchas de ellas solo son un sistema de recomendación de código cerrado y no permiten reutilizar sus funciones de esta forma para algún desarrollador que desee realizar una aplicación similar.

3.1. Spotify

Desarrollado en el 2008, es un software multiplataforma que permite escuchar en modo radio buscando por artista, álbum o listas de reproducción creadas por los propios usuarios. Spotify hace uso de un sistema de recomendación para mostrar canciones y listas de reproducción a sus usuarios. Utiliza una transferencia de archivos de audio por Internet a través de la combinación de servidor basado en el streaming y en la transferencia peer-to-peer (P2P). Cuenta con versiones para Windows, Mac, Linux y una versión web. [4]

3.2. Trabajo Terminal Sistema Generador de Recomendaciones para una Tienda En-Línea de videojuegos.

Desarrollado en el 2010, es una plataforma web de información para la venta de artículos de entretenimiento. Este Trabajo Terminal no es una API pero utiliza un sistema de recomendaciones para mejorar las compras de sus usuarios. Palabras clave: Recomendación, TopN, mercancía, tienda en línea.[5]

3.3. PredictionIO

Proporciona los recursos necesarios para crear un servidor de recomendaciones usando aprendizaje máquina. Todo a través de una API REST que se comunica con las distintas aplicaciones clientes y va recolectando datos para aplicar los más de 20 algoritmos de recomendaciones precargados. Cuenta con varios SDK para integrarlo en sus aplicaciones como Java, Python, Ruby o PHP. Además de sistemas de recomendación y aprendizaje máquina Plataformas: Linux/MacOSX, Vagrant(VirtualBox), Web, Source Code.[6]

3.4. Easyrec

Sistema de personalización de propósito general, utiliza una API de servicio web actual está personalizada para proporcionar tiendas en línea con recomendaciones de artículos. Cabe mencionar que es una aplicación lista para usar, no un framework algorítmico. No es una API, pero si hace eso de una API el cual hace las recomendaciones. Componentes: aplicación para almacenar servicios de recomendación, API para varias interfaces de servicio web. Año: 7 de mayo 2013[7]

3.4.1. REST API

API easyrec es capaz de recibir acciones desde o enviar recomendaciones a la aplicación web en estilo REST. Todas las acciones de los usuarios enviados a easyrec son analizados por el ruleminer, que se ejecuta una vez al día de forma predeterminada. Si similitudes significativas en el comportamiento del usuario se pueden encontrar, se generan recomendaciones y son accesibles al instante utilizando los servicios web de recomendación. Componentes: XML o notación JSON para mostrar las recomendaciones. [8]

3.5. LensKit

Desarrollado principalmente por la Universidad del Estado de Texas y por GroupLens Research en la Universidad de Minnesota, es una API para desarrollo de sistemas de recomendación, que funciona bajo algoritmos de recomendación para artículos y usuarios para realizar las recomendaciones de los artículos. Su implementación puede ser observada en el sistema de recomendación de películas MovieLens desarrollado por el mismo equipo de trabajo. [11]

Capítulo 4

Marco teórico

4.1. Sistemas de recomendación

Frecuentemente es necesario hacer elecciones sin tener la suficiente experiencia personal o conocimiento sobre las alternativas existentes. En la vida diaria, se confía en las recomendaciones que otras personas hacen de palabra, cartas de recomendación, reseñas de libros, películas o encuestas generales. [4] Los sistemas de recomendación asisten y aumentan este proceso social. En un sistema de recomendación típico la gente provee de información como entradas que sirven para dar recomendaciones y cabe destacar que se diferencia de un motor de búsqueda por la razón de individualidad que se pretende brindar al tener un sistema personalizado, ya que el criterio de qué tan relevante o útil pueda ser un artículo varía para cada usuario y no es general para todos. Estas diferencias generan distintas aproximaciones en el comportamiento de un sistema de recomendación, donde la tendencia es combinar distintas técnicas de recomendación para mejorar el desempeño del sistema. Las técnicas de recomendación tienen distintas clasificaciones dependiendo de las fuentes de datos en las que se basa la recomendación y el uso en que la información es puesta. Específicamente, los sistemas de recomendación tienen (i) información previa, la información que el sistema tiene antes de que se realice la recomendación; (ii) información de entrada, la información que el usuario debe comunicar al sistema para generar una recomendación, y (iii) un algoritmo que combina la información previa y la recibida del usuario para mostrar sus sugerencias. Con esta base, es posible identificar diferentes técnicas.

4.1.1. Técnicas de recomendación

Las recomendaciones *colaborativas* es probablemente la más familiar, mayormente implementada y más madura de las tecnologías. Los sistemas de recomendación colaborativos agregan ratings (evaluaciones) o recomendaciones de objetos, reconocen la similitud entre los usuarios de acuerdo a sus ratings y generan recomendaciones con base en las comparaciones entre usuarios. Las recomendaciones *demográficas* pretenden categorizar al usuario con base en sus atributos personales y hacer recomendaciones de acuerdo a clases demográficas. La representación de la información demográfica en el modelo puede variar enormemente, dependiendo de las necesidades del sistema y de la información que pueda recabar. Las técnicas de recomendación demográficas también utilizan algún tipo de correlación entre los usuarios, pero difieren de las colaborativas en medida de que no requieren de la interacción explícita de los ratings para obtener recomendaciones. Las recomendaciones *basadas en contenido* representan la continuación y crecimiento de las investigaciones sobre filtrado

de datos. En un sistema de recomendación basado en contenido, los objetos de interés son definidos por sus características que pueden ser asociadas entre varios objetos. El sistema basado en contenido utiliza la información de los objetos para categorizarlos a través de una correlación entre los objetos y entonces, con base en la interacción del usuario, recomendar objetos similares a los que ha visualizado. Los perfiles de usuarios dependen del método de aprendizaje empleado. Han sido usados árboles de decisión, redes neuronales y representaciones basadas en vectores por igual. Este tipo de modelos son de larga duración y solo son actualizados mientras el usuario tiene mayor interacción con los mismos. Por el contrario, las recomendaciones *por utilidad* y *basadas en conocimiento* no pretenden dar soluciones generales a largo plazo sobre sus usuarios, sino que se basan en la evaluación entre la necesidad actual del usuario y el set de opciones disponible. Las recomendaciones basadas en utilidad hacen sugerencias con base en el cómputo de la utilidad que tiene el usuario para cada objeto. Por supuesto, el problema central es como crear una función de utilidad para cada usuario. El beneficio de las recomendaciones basadas en utilidad es que permiten factorizar atributos, haciendo posible elevar la importancia de una característica frente a otra. Los sistemas *basados en conocimiento* pretenden sugerir objetos basados en inferencias acerca de las necesidades del usuario y preferencias. En cierto sentido, se puede decir que todas las técnicas de recomendación hacen algún tipo de inferencia, pero los sistemas basados en conocimiento se distinguen porque tienen conocimiento funcional, de como un artículo en particular puede empatar una necesidad de usuario. El modelo para el perfil de usuario puede ser cualquier estructura de conocimiento que permita la inferencia. [5]

4.1.2. Comparando técnicas de recomendación

Todas las técnicas de recomendación tienen sus fortalezas y sus debilidades, quizás el más conocido es el problema de ramp-up, que se aplica tanto para usuarios como para objetos. Ya que al tener un usuario nuevo, es difícil de categorizar debido a la poca o nula interacción que tenga con el sistema. De la misma manera, cuando un objeto es nuevo, no tiene muchas evaluaciones de usuarios y no puede ser recomendado tan fácilmente. Entre estos, las principales ventajas y desventajas de las técnicas de recomendación mencionadas pueden ser visualizadas en la tabla 4.1[5]

4.1.3. Sistemas híbridos

Con el fin de obtener un mejor desempeño en los sistemas de recomendación, se utilizan al mismo tiempo dos o más de las técnicas descritas previamente para formar lo que se denomina un sistema híbrido, donde estas diferentes técnicas pueden convivir de maneras diferentes como se puede visualizar en la tabla 4.2

Técnica	Fortalezas	Debilidades
Filtrado colaborativo	<ul style="list-style-type: none"> ■ Puede identificar nichos cruzados ■ No necesita dominio de conocimiento ■ Adaptativo: la calidad mejora sobre el tiempo ■ Retroalimentación implícita suficiente 	<ul style="list-style-type: none"> ■ Problema ramp-up con los usuarios nuevos ■ Problema ramp-up con los objetos nuevos ■ Problema de la oveja gris (problemas con clasificaciones que no pertenecen en concreto a un nicho) ■ La calidad depende de grandes volúmenes de históricos. ■ El problema de la plasticidad contra la estabilidad.
Basado en contenido	<ul style="list-style-type: none"> ■ No necesita dominio de conocimiento ■ Adaptativo: la calidad mejora sobre el tiempo ■ Retroalimentación implícita suficiente 	<ul style="list-style-type: none"> ■ Problema ramp-up con los usuarios nuevos ■ La calidad depende de grandes volúmenes de históricos. ■ El problema de la plasticidad contra la estabilidad.
Filtrado demográfico	<ul style="list-style-type: none"> ■ Puede identificar nichos cruzados ■ No necesita dominio de conocimiento ■ Adaptativo: la calidad mejora sobre el tiempo 	<ul style="list-style-type: none"> ■ Problema ramp-up con los usuarios nuevos ■ Problema de la oveja gris (problemas con clasificaciones que no pertenecen en concreto a un nicho) ■ La calidad depende de grandes volúmenes de históricos. ■ El problema de la plasticidad contra la estabilidad. ■ Debe obtener información demográfica sensible.

Técnica	Fortalezas	Debilidades
Basado en utilidad	<ul style="list-style-type: none"> ■ No requiere curva de aprendizaje (ramp-up) ■ Sensible a los cambios en las preferencias ■ Puede incluir características que no son de los productos 	<ul style="list-style-type: none"> ■ El usuario debe introducir la función de utilidad ■ Habilidad de sugerencia estática (no aprende)
Basado en conocimiento	<ul style="list-style-type: none"> ■ No requiere curva de aprendizaje (ramp-up) ■ Sensible a los cambios en las preferencias ■ Puede incluir características que no son de los productos ■ Puede mapear de necesidad de usuario a productos 	<ul style="list-style-type: none"> ■ Habilidad de sugerencia estática (no aprende) ■ Requiere ingeniería de conocimiento.

Cuadro 4.1: Comparación de las distintas técnicas de recomendación

Método de hibridización	Descripción
Ponderado	Las evaluaciones (o votos) de distintas técnicas de recomendación son combinadas juntas para producir una sola recomendación.
Switching	El sistema intercambia entre técnicas de recomendación dependiendo de la situación.
Mezclado	Recomendaciones de diferentes técnicas son presentadas al mismo tiempo.
Combinación de características	Características de distintas fuentes de información son utilizadas juntas en un solo algoritmo de recomendación
Cascada	Una técnica de recomendación refina las recomendaciones dadas por otra.
Aumento de funcionalidad	La salida de una técnica es usada como entrada de otra.
Nivel-meta	El modelo aprendido por un recomendador es usado como entrada de otro.

Cuadro 4.2: Sistemas híbridos

[5]

4.2. Algoritmos de recomendación

4.2.1. Vecinos próximos (k-NN)

Este método selecciona los k vecinos más similares al usuario o al ítem objetivo, de forma que mediante la combinación lineal del rating de los vecinos se realiza una predicción de rating. A partir de esta predicción, podemos sacar el ránking de ítems a recomendar, simplemente ordenándolos por orden descendente del rating predicho. Existe una gran cantidad de variaciones para estos algoritmos, ya que tienen muchos parámetros configurables que dan diferentes resultados dependiendo de los datos con los que se traten. El número k de vecinos, la similitud mínima a considerar como relevante, o el número de ítems en común para una similitud válida son algunos de los parámetros a establecer para estos algoritmos. Los algoritmos de vecinos próximos se han desarrollado en dos perspectivas posibles: recomendación de vecinos próximos por usuario y por ítem.

Basado en usuario

Se recomiendan al usuario los ítems que han gustado a usuarios similares a éste. Su fórmula es la siguiente:

$$f(u, i) = \sum_{\substack{v \in N_k(u) \\ r(v, i) \neq \emptyset}} sim(u, v) * r(v, i)$$

Basado en ítem

Se recomiendan al usuario los ítems que se parecen a ítems que le han gustado. Su función es muy similar a la del algoritmo k-NN con usuarios.

$$f(u, i) = \sum_{\substack{j \in N_k(i) \\ r(u, j) \neq \emptyset}} sim(i, j) * r(u, j)$$

Normalmente, este algoritmo se utiliza con el número de vecinos igual al número total de ítems, es decir, que el vecindario de cada ítem son todos los demás existentes en el conjunto de datos. En cuanto a la función de similitud entre los usuarios o ítems, ésta se puede realizar mediante varios métodos, entre ellos destacan:

- **Coseno:** Mide el ángulo entre los vectores (ratings) de cada par de usuarios o ítems, de forma que son más similares aquellos cuyos vectores tienen la misma orientación.

$$sim(u, v) = \frac{\sum_{i: r(u, i) \neq \emptyset, r(v, i) \neq \emptyset} r(u, i) * r(v, i)}{\sqrt{\sum_{i: r(u, i) \neq \emptyset} r(u, i)^2} \sqrt{\sum_{i: r(v, i) \neq \emptyset} r(v, i)^2}}$$

- **Jaccard:** Dos usuarios o ítems son más similares cuanto más parecida sea la intersección a la unión de ambos, es decir, cuantos más ratings en común tengan, independientemente del valor de rating.

$$sim(u, v) = \frac{|u \cap v|}{|u \cup v|} = \frac{|u \cap v|}{|u| + |v| - |u \cap v|}$$

- **Pearson:** Equivalente a la similitud por coseno, pero cada rating se centra en la puntuación promedio del usuario (o ítem) correspondiente. Para este caso existen dos versiones dependiendo de la forma de calcular el módulo de v : por intersección, donde se suman los ratings de los ítems que tiene en común con u ; y total, donde también se incluyen los ratings de aquellos que u desconoce.

$$sim(u, v) = \frac{\sum_{i: r(u,i) \neq \emptyset} (r(u, i) - \bar{r}_u) (r(v, i) - \bar{r}_v)}{\sqrt{\sum_{i: r(u,i) \neq \emptyset} (r(u, i) - \bar{r}_u)^2} \sqrt{\sum_{i: r(v,i) \neq \emptyset} (r(v, i) - \bar{r}_v)^2}}$$

4.2.2. Slope One

El filtrado colaborativo es una técnica usada por los Sistemas de Recomendación para combinar las opiniones y pruebas de diferentes usuarios con el fin de obtener recomendaciones personalizadas. Hay al menos dos clases de filtrados colaborativos: las técnicas basadas en usuarios son derivadas de la medición de similitudes entre usuarios, mientras que las técnicas basadas en artículos comparan las valoraciones dadas por distintos usuarios. Slope One es una familia de algoritmos usados para el Filtrado Colaborativo introducida en Slope One Predictors for Online Rating-Based Collaborative Filtering por Daniel Lemire y Anna Maclachlan. Posiblemente, esta es la forma más simple de filtrado colaborativo basado en artículos. Su simplicidad la hace especialmente sencilla de implementar eficientemente mientras que su exactitud está a la par de algoritmos más complejos y costosos.

4.2.3. Factorización de matrices

A los gustos de los usuarios y las características de los ítems subyace un espacio no visible que realmente determina por qué les gustan los ítems. Hay formalismos matemáticos que nos permiten hacer aflorar este tipo de espacio latente como unas cuantas dimensiones reducidas, sin tener que concretar qué son realmente esas dimensiones, y pudiendo sin embargo generar recomendaciones operando con ellas. La idea es representar tanto los ítems como los usuarios como vectores en un espacio de factores latentes, con una coordenada por factor que representa el grado de afinidad del usuario (o del ítem) hacia ese factor. Este efecto se puede realizar mediante la factorización de matrices representada en la figura 4.1, descomponiendo la matriz original de ratings en un producto de varias matrices, dos o tres dependiendo del algoritmo, obteniendo siempre una primera matriz de usuarios por factores y una última de factores por ítems.

De esta forma, se obtienen k factores latentes que establecen un espacio de características común, tanto para los usuarios como para los ítems, permitiendo la comparación directa entre ellos. Así, un usuario da ratings de acuerdo a sus factores latentes y a los factores latentes del ítem. A partir de aquí se ha desarrollado en el área gran cantidad de algoritmos para obtener la factorización de matrices, entre los que destacamos los siguientes:

pLSA(probabilistic Latent Semantic Analysis): Divide la matriz de ratings en dos, usuarios por factores y factores por ítems, de forma que su función de ránking equivale

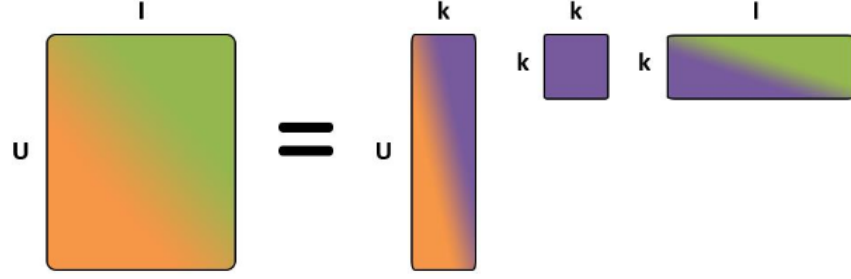


Figura 4.1: Descomposición de la matrix en el producto de otras tres

a la probabilidad de que el usuario puntúe al ítem según el espacio de factores latentes.

$$f(u, i) = p(i|u; \theta) = \sum_f p(i|f) * p(f|u)$$

SVD(Singular Value Decomposition): La factorización de la matriz de ratings genera tres matrices, para lo que se tiene en cuenta todos los ratings, asignándoles un cero a aquellos no conocidos. Se obtienen los vectores de usuario e ítem mediante el producto de las matrices obtenidas según la fórmula que se indica a continuación, de forma que su función de ránking consiste en la multiplicación de los vectores de usuario e ítem, centrado en la media del usuario.

$$r = W * S * F^T$$

$$f(u, i) = \bar{r}_u + W\sqrt{S}^T(u) * \sqrt{S} F^T(i)$$

SVDN(SVD no-empty entries): Variante del anterior algoritmo en la que se obtienen dos matrices en lugar de tres, se tiene en cuenta únicamente los ratings conocidos, y su función de ránking no se centra en la media del usuario.

$$r = U_F * I_F$$

En este caso, entonces, el cálculo de dicha función es más inmediato, ya que los vectores de usuario e ítem se corresponden con la fila de la primera matriz y la columna de la segunda respectivamente, por lo que únicamente se multiplican la fila por la columna correspondiente.

$$f(u, i) = U_F(u) * I_F(i)$$

HSVD(SVD with Hypergraph transformation): Este método está dirigido a la recomendación de ítems a usuarios nuevos en el sistema, es decir, con poca actividad en el entrenamiento. Por ello, el primer paso del algoritmo consiste en dividir la matriz de ratings en tres: ratings de los usuarios que no están en test, ratings conocidos (de entrenamiento) de los usuarios que están en test, y ratings desconocidos de los usuarios que están en test (lo que se recomendará).

ASVD(Assymetric SVD): Se trata de la versión asimétrica de SVD, que solo usa la tercera matriz de la descomposición para realizar las recomendaciones. Sigue exactamente los mismos pasos que HSVD, con la diferencia de que no se binariza la matriz de ratings de los usuarios viejos, sino que se utiliza la original con los ratings numéricos.

4.2.4. Basados en contenido

Este grupo de algoritmos se basa en la utilización de la descripción de cada ítem para recomendar, sin utilizar información de otros usuarios para generar la recomendación al usuario objetivo. Destacan Rocchio y la implementación del k-NN para los ítems. . **Rocchio:** Se basa en el cálculo de centroides para cada usuario, de forma que se obtenga un vector representante para cada uno. Estas clases se corresponderán con las características de los ítems, por ejemplo, en Twitter, las palabras clave del contenido de los tweets. De esta forma, se obtiene para cada usuario un centroide que representa su relación con cada característica. La fórmula para el cálculo de los centroides es la siguiente:

$$u[f] = \frac{1}{|u|} \sum_{i:r(u,i) \neq \emptyset} tfidf(f, i) * r(u, i), \text{ donde } u = \{r(u, i) \neq \emptyset | i \in \mathcal{I}\}$$

Una vez se dispone de los centroides, el cálculo de la similitud de los usuarios con cada uno de los ítems se realiza mediante cualquiera de los métodos anteriormente descritos.

k-NN (ítems): Para la implementación del k-NN, la estructura de este algoritmo es idéntica a la de filtrado colaborativo del mismo nombre, pero se diferencian en la forma de calcular la similitud entre los ítems.

$$sim(i, j) = \frac{\sum_f tfidf(f, i) * tfidf(f, j)}{\sqrt{\sum_f tfidf(f, i)^2} \sqrt{\sum_f tfidf(f, j)^2}}$$

4.2.5. Algoritmos no personalizados

Estos algoritmos recomiendan ítems sin conocer ningún dato del usuario, de forma impersonal como su propio nombre indica.

Popularidad: Recomienda los ítems por orden de popularidad y, por tanto, da exactamente el mismo ranking a cada uno de los usuarios. Por popularidad de un ítem se entiende el número de usuarios que han interactuado con el ítem en el sistema. Este método puede parecer trivial, pero es sin embargo uno de los más extendidos en escenarios reales.

Random: Recomienda ítems de manera aleatoria a cada usuario y su precisión está relacionada con la densidad del conjunto de datos. La efectividad de la recomendación aleatoria, medida con una cierta métrica, se puede interpretar como la esperanza del valor de la métrica sobre el conjunto de datos en el que se aplica. [11]

4.3. Bases de datos orientadas a grafos

Las bases de datos relacionales han existido por muchas décadas y son la tecnología de base de datos de elección para la mayoría de almacenamientos intensivos de datos y aplicaciones de obtención de datos. La obtención generalmente se logra mediante SQL, un lenguaje declarativo de consultas. Los sistemas de bases de datos relacionales son generalmente eficientes a menos que la información contenga muchas relaciones que requieren la unión de tablas grandes. Recientemente ha habido mucho interés en los almacenes de datos que no utilizan SQL exclusivamente, el llamado movimiento NoSQL. Ejemplos de ello son BigTable de Google y Cassandra de Facebook. [7]

Los modelos de bases de datos orientados a grafos se pueden caracterizar como aquellos en los que las estructuras de datos para el esquema y los casos se modelan en forma de grafos o generalizaciones de ellos, y la manipulación de datos se expresa mediante operaciones orientadas a grafos. Estos modelos nacieron en los años ochenta y principios de los noventa

en paralelo a los modelos orientados a objetos y su influencia se desvaneció poco a poco con la aparición de otros modelos de bases de datos, en particular la geográfica, espacial, semi-estructurada y XML.

Recientemente, la necesidad de gestionar la información con naturaleza inherente de un grafo ha hecho volver la relevancia de la zona. De hecho, una nueva ola de aplicaciones para bases de datos de grafos surgió con el desarrollo de las redes grandes (por ejemplo, Web, sistemas geográficos, transporte, teléfonos), y familias de redes generadas gracias a la automatización del proceso de recopilación de datos (por ejemplo, redes sociales y biológicas (incluyendo redes metabólicas, redes de interacción entre proteínas, grafos de estructuras químicas, clústers de genes, y mapas genéticos. Los grafos son realmente una de las más útiles estructuras para modelar objetos e interacciones.[8]

4.3.1. Características

No hay índices clásicos en las bases de datos basadas en grafos. Por el contrario, cada objeto almacenado es representado por nodos (entidades) y aristas (relaciones). Un nodo es un registro único que tiene al menos una propiedad. Las aristas definen las relaciones entre los nodos y los nodos y sus relaciones tienen a su vez predefinidas conjuntamente propiedades. Los nodos pueden tener múltiples aristas que definen los diferentes tipos de relaciones que tienen con otros nodos.

Las consultas en las bases de datos orientadas a grafos están diseñadas para empezar en un nodo específico y explorar sus relaciones con otros nodos. Un ejemplo podría ser: ¿Qué libros están leyendo mis amigos que yo aún no haya leído? Es por eso que este tipo de bases de datos están frecuentemente asociadas con motores de recomendación que se usan con frecuencia en aplicaciones sociales y de comercio electrónico.

A medida que las búsquedas se van haciendo más complejas, el tiempo de procesamiento va aumentando. Es por eso que las bases de datos basadas en grafos aprenden e indexan las relaciones más comunes con el objetivo de acelerar el tiempo de búsqueda.

Ventajas

- Rapidez para conectar datos. En las bases de datos relacionales, el frecuente uso de joins hace que las búsquedas sean lentas.
- Sencillez de las consultas.
- Rapidez en el manejo de consultas complejas que implican múltiples niveles de datos relacionados.

Desventajas

- La búsqueda de nodos en diferentes máquinas puede ralentizar el proceso drásticamente.
- Requiere un cambio conceptual para los desarrolladores, por lo que implica una curva de aprendizaje.

[9]

4.4. Application Programming Interface (API)

Una API (siglas de ‘Application Programming Interface’) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

Las API pueden servir para comunicarse con el sistema operativo (WinAPI), con bases de datos (DBMS) o con protocolos de comunicaciones (Jabber/XMPP). En los últimos años, por supuesto, se han sumado múltiples redes sociales (Twitter, Facebook, Youtube, Flickr, LinkedIn, etc) y otras plataformas online (Google Maps, WordPress...), lo que ha convertido el social media marketing es algo más sencillo, más rastreable y, por tanto, más rentable.

Las API son valiosas, ante todo, porque permiten hacer uso de funciones ya existentes en otro software (o de la infraestructura ya existente en otras plataformas) para no estar reinventando la rueda constantemente, reutilizando así código que se sabe que está probado y que funciona correctamente. En el caso de herramientas propietarias (es decir, que no sean de código abierto), son un modo de hacer saber a los programadores de otras aplicaciones cómo incorporar una funcionalidad concreta sin por ello tener que proporcionar información acerca de cómo se realiza internamente el proceso. [3]

Eso es cierto incluso para los programas de código abierto. ¿Quién tiene el tiempo para revisar todo el código de la aplicación de otra persona, el cual puede ser terriblemente desordenado, sólo para usar una función? Las API simplifican todo al limitar el acceso fuera de programa a un conjunto específico de características, a menudo suficientes. Se definen claramente exactamente cómo un programa que va a interactuar con el resto del software, ahorrar recursos y enredos legales potencialmente desagradables en el camino. [10]

Capítulo 5

Análisis general del proyecto

Para el desarrollo del sistema en su totalidad, se realizó el siguiente análisis general que define en su totalidad al proyecto.

5.1. Características

Para que el sistema se considere que ha cumplido con los objetivos planteados debe contar con las siguientes características dentro de su funcionalidad.

- El sistema permitirá la búsqueda de platillos.
- El sistema permitirá obtener recomendaciones de acuerdo a las características de un platillo.
- El sistema obtendrá información de la interacción del usuario a través de evaluaciones explícitas, y conteo de clics implícitos hacia los platillos para obtener recomendaciones personalizadas para dicho usuario.
- El sistema permitirá el registro de usuarios finales.
- El sistema generará recomendaciones de platillos de acuerdo a la información proporcionada de los usuarios registrados, con base en sus evaluaciones y características.
- El sistema permitirá al usuario agregar nuevos platillos que haya consumido en cierto restaurante.
- El sistema permitirá al usuario administrar los platillos que agregó.

5.2. Restricciones

Debido a diferentes aspectos, el sistema contará con las siguientes restricciones.

- El sistema se verá limitado a las tecnologías empleadas para su desarrollo.
- El sistema puede verse limitado debido a la dependencia de fuentes de información externas.
- El sistema puede verse limitado en desempeño y precisión debido a la cantidad de información almacenada y a la complejidad del problema a resolver.

5.3. Estudio de factibilidad

Después de definir la problemática presente y presentar la propuesta de solución, es pertinente realizar un estudio de factibilidad para determinar la infraestructura tecnológica y la capacidad técnica que implica la implementación de la API en cuestión. Este análisis permite determinar las posibilidades de diseñar a API propuesta y su puesta en marcha.

A continuación se describen los aspectos que se toman en cuenta para este análisis.

5.3.1. Factibilidad técnica

Consiste en realizar una evaluación de la tecnología existente; este estudio está destinado a recolectar información sobre los componentes tecnológicos que posee este equipo de desarrollo y la posibilidad de utilizarlos en el desarrollo e implementación de este trabajo y de ser necesario, los requerimientos tecnológicos que deben ser adquiridos su desarrollo e implementación.

Hardware

Para el desarrollo de la API se cuentan con 3 computadoras personales, las cuales cuentan con las siguientes características:

- MacBook Pro 13: 4 GB DDR3 RAM, Procesador Intel i5 a 2.5 GHz. S.O
- HP 8 GB DDR3 RAM, Procesador A10 2.1 GHz. S.O. Ubuntu 14.04.
- Acer 8 GB DDR3 RAM Procesador AMD A6

Los módulos desarrollados para la primera parte del Trabajo Terminal serán expuestos de manera local, accediendo a ellos a través de una red local. Para la segunda entrega del Trabajo Terminal se tiene planeado exponerlos en un servidor, para lo cual se contará un servicio de hosting. No se incluirá en este documento la opción elegida para el hosting ni el costo del mismo, ya que no se sabe con exactitud cual se elegirá, a pesar de que se han tomado en cuenta varios, no se ha llegado a una decisión.

Software

- Java
- Neo4j
- Hibernate
- JavaScript
- HTML
- Bootstrap
- Spring
- Maven

Estas tecnologías son necesarias para el desarrollo de esta API. Cada una cumple con un objetivo específico para las cuales fueron utilizadas, pero no son indispensables. No son indispensables, ya que se encuentran muchos otros lenguajes y frameworks con los cuales se pueden reemplazar estas tecnologías.

Costos

Los costos que generará el desarrollo de este framework se calcularon de la siguiente manera:

- El manejo roles se repartió en el equipo, es decir, todos tuvieron que analizar, diseñar y desarrollar.
- Nos basamos en un sueldo promedio al cual aspiran estudiantes de la carrera de Ing. en Sistemas Computacionales, que es de \$ 80

Después de aclarar lo siguiente, los costos operacionales (mano de obra) se calcularon así:

- 3 personas que desarrollan diferentes actividades
- Cada uno gana \$80 la hora desarrollando la API
- Se toma en cuenta que se trabajan los 7 días de la semana 4 horas cada uno de ellos, así:

$$(\$80) * (4 \text{ hrs}) * (250 \text{ dias}) = \$80,000$$

El precio estimado de este sistema es de \$80000 mx solamente tomando en cuenta la manos de obra, sin contemplar nuevos equipos, reuniones, transportes, comida y horas extras.

5.4. Metodología

5.4.1. Descripción

Para ese proyecto se plantea usar una metodología de prototipado evolutivo, que se caracteriza por que en su modelo de trabajo un prototipo es construido, probado y finalmente reconstruido las veces que sea necesario hasta que un prototipo aceptable es finalmente alcanzado del cual el sistema completo o producto puede ser totalmente desarrollado.

Este modelo funciona bien en escenarios donde no se conocen por completo los requerimientos. Así los prototipos, modelos de software con una funcionalidad limitada, permiten al usuario evaluar los propósitos del desarrollador y probarlos antes de su implementación. También ayuda a entender los requerimientos específicos del usuario y que no pudieron haber sido considerados por el desarrollador durante el diseño del sistema.

5.4.2. Prototipos esperados

De acuerdo al modelo de desarrollo elegido, se han establecido los siguientes prototipos y sus alcances que se pueden observar en el cuadro 5.1

Versión	Alcance
Prototipo 1	Obtener y abstraer los datos a utilizar
Prototipo 2	Visualizar los datos en recomendaciones por contenido
Prototipo 3	Obtener recomendaciones para diversos fines (sistemas híbridos)
Prototipo 4	Desarrollar un sistema híbrido de recomendación para platillos y restaurantes.

Cuadro 5.1: Cuadro de prototipos esperados

5.5. Descripción y módulos del sistema

5.5.1. Arquitectura general

Para desarrollar un sistema de recomendación se han planteado los siguientes módulos funcionales de la API, el cual será utilizado por el desarrollador final para que, en conjunto con su aplicación final realice la integración de las funciones disponibles en el API junto al sistema de recomendación final. Esto se puede denotar en el diagrama de la figura 5.1

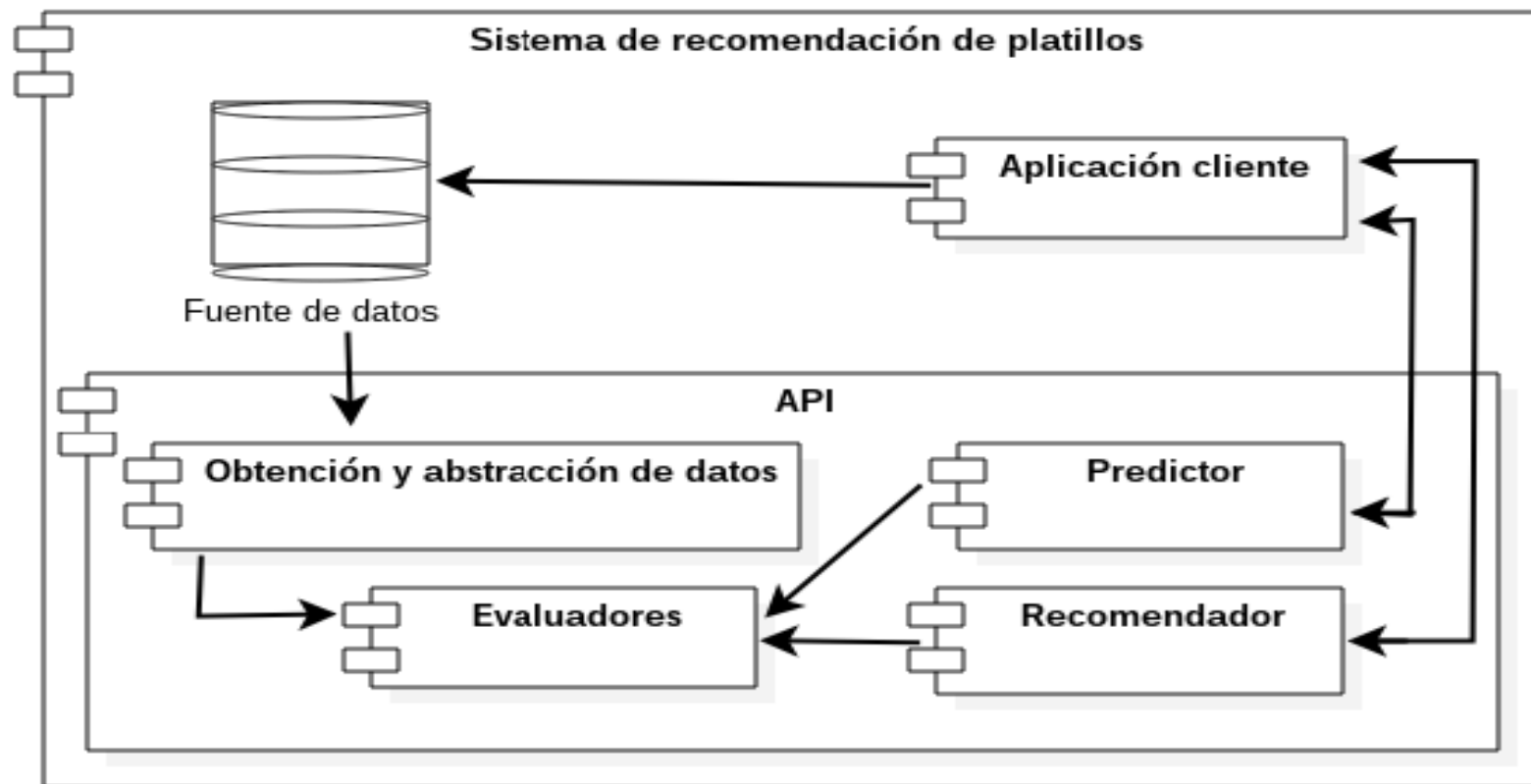


Figura 5.1: Diagrama general del sistema

Cómo se puede apreciar en el diagrama, el sistema se encuentra dividido en tres módulos básicos además de la aplicación final que hace uso de los módulos de la API.

- Módulo de obtención datos
- Módulo de evaluación
- Módulo de presentación de resultados
- Aplicación cliente

5.5.2. Módulos de la API

Módulo de obtención de datos

Este módulo es el encargado de la conexión a una fuente de datos de la cuál obtendrá los datos a utilizar, mapeándolos dentro de una estandarización propuesta por el equipo de trabajo. Esta obtención permitirá tener la funcionalidad del sistema de recomendación de manera adecuada. Se presenta como un módulo conformado por diferentes interfaces de conexión para fuentes de datos, como lo es un gestor de base de datos. Tiene una interacción directa con el módulo de operaciones lógicas dentro del sistema. Dentro de sus características podemos denotar:

- Permite la conexión a una fuente de datos.
- A través de interfaces, permite mapear los datos utilizados en cada caso de estudio para el correcto funcionamiento del módulo de operaciones lógicas.
- Se encuentra restringido al modelo de datos mínimo propuesto por el equipo de trabajo.

Módulo de operaciones lógicas

El módulo de operaciones lógicas permitirá, haciendo uso de la información almacenada, obtener evaluaciones de los diferentes artículos de acuerdo a diferentes clasificaciones con base en algoritmos de recomendación que definen los principales tipos de recomendación existentes: basados en contenido, colaborativos e híbridos. Estas evaluaciones deberán ser utilizadas por el módulo de presentación de resultados para devolver predicciones o recomendaciones que estén dentro de los rangos de evaluación propuestos por el desarrollador para su caso de estudio. Cabe destacar que este tipo de algoritmos devolverán evaluaciones cuantitativas, y la representación de estos se definirá dependiendo el caso, siendo lo ideal que a un mayor valor corresponda una mayor similitud o preferencia. Dentro de sus características principales encontramos:

- Hace uso de la conexión proporcionada por el módulo de datos y los parámetros indicados en el módulo de presentación de resultados para devolver las evaluaciones.
- Las evaluaciones son cuantitativas, denotando idealmente una mayor relevancia para el usuario como un mayor valor.
- El algoritmo de evaluación a utilizar será determinado en la interacción entre éste módulo y el módulo de presentación de resultados.

Módulo de presentación

Haciendo uso del módulo de operaciones lógicas, este módulo pretende interactuar con la aplicación final para brindar las funciones de recomendación y predicción para cada usuario o artículo.

5.5.3. Aplicación final

Definición

En este caso, la aplicación final hará uso de las funciones proporcionadas por los distintos módulos que conforman la API para obtener recomendaciones de los datos que pertenezcan a su caso de estudio. Interactúa directamente con el módulo de abstracción de datos y con el módulo de presentación para hacer uso de la funcionalidad permitida por el API. En este caso, la aplicación final se verá reflejada en un sistema web que permita denotar la funcionalidad de la API para un conjunto de datos de platillos y restaurantes.

5.6. Análisis y gestión de riesgos

5.6.1. Definición y clasificación

El riesgo siempre implica una incertidumbre y una pérdida potencial, al identificar estos riesgos podemos determinar su naturaleza en tres tipos diferentes para este proyecto:

- Riesgos del proyecto
- Riesgos técnicos
- Riesgos de negocio

Así mismo, es necesario clasificar los riesgos existentes de acuerdo a la probabilidad de que éstos ocurran. Para lo cual se utilizará los siguientes valores por convención.

- Muy bajo (<10 %)
- Bajo (10 - 25 %)
- Moderado (25 - 50 %)
- Alto (50 - 75 %)
- Muy Alto (>75 %)

Así mismo, todos los riesgos deben categorizarse de acuerdo al impacto que pueden causar en el sistema en las siguientes categorías: Insignificante, Tolerable, Serio, Catastrófico; los planes de contingencia suelen ser desarrollados para aquellos riesgos con probabilidad de moderada a muy alta y con un impacto serio o catastrófico. A continuación en el cuadro 5.2 se plantean los riesgos identificados para el sistema general a lo largo del desarrollo del mismo y sus respectivos planes de acción.

Tabla de Riesgos				
Descripción	Tipo de Riesgo	Valoración	Porcentaje	Plan de acción
Falta de presupuesto	Proyecto	Serio	25 %	Buscar un proceso de incubación en empresas como Apache, Eclipse y migrar la plataforma a servicios de hosting gratuitos como Heroku u Openshift.
Falta por razones personales de miembros del equipo	Proyecto	Serio	25 %	Rediseñar y adaptar las tareas con nuevos integrantes y habilitar forma de trabajo remota considerando fines de semana.
Necesidad de escalabilidad de la plataforma	Técnico	Serio	10 %	Definir el tipo de escalabilidad a usar dependiendo los recursos monetarios existentes.

Cuadro 5.2: Analisis de Riesgos

5.7. Tecnologías

Para el desarrollo del sistema, existen varias tecnologías en el mercado que cumplen con el propósito de desarrollar el sistema y alcanzar los objetivos planteados. Para esto es necesario tener el conocimiento acerca de las características que pueden brindar las tecnologías que tenemos a nuestro alcance y hacer una elección sobre cuales ofrecen mayores ventajas a la hora de desarrollar el sistema.



Cuadro comparativo de tecnologías	
Nombre	Características
Base de datos relacional	<ul style="list-style-type: none">■ Es un tipo de base de datos que cumple con el modelo relacional.■ Permite establecer interconexiones o relaciones entre los datos, y a través de dichas conexiones relacionar los datos de ambas tablas, de ahí proviene su nombre: "modelo relacional".■ No pueden existir dos tablas con el mismo nombre ni registro.■ Cada tabla es a su vez un conjunto de campos (columnas) y registros (filas).■ Las claves primarias son la clave principal de un registro dentro de una tabla y estas deben cumplir con la integridad de datos.[26]
Bases de datos no relacional	<ul style="list-style-type: none">■ Los datos almacenados no requieren estructuras fijas como tablas, normalmente no soportan operaciones JOIN.■ Pueden manejar enormes cantidades de datos.■ No generan cuellos de botella.■ Escalamiento sencillo.■ Se ejecutan en clústers de máquinas baratas.■ No usan SQL como el principal lenguaje de consultas [27]

Base de datos Orientada a grafos	<ul style="list-style-type: none"> ■ Representa la información como nodos de un grafo y sus relaciones con las aristas del mismo ■ Consultas más amplias y no demarcadas por tablas. ■ No hay que definir un número determinado de atributos. ■ Los registros también son de longitud variable, evitando tener que definir un tamaño y también posibles fallas en la base de datos. ■ Se puede recorrer directamente la base de datos de forma jerárquica, obtener el nodo abuelo del nodo y viceversa. [28]
----------------------------------	---



Cuadro 5.3: Tipos de bases de datos

Cuadro comparativo de tecnologías	
Nombre	Características
	<ul style="list-style-type: none"> ▪ Es una base de datos open-source, esta escrita en java. ▪ Permite realizar transacciones ACID. ▪ Manera su propio lenguaje de query , Cypher. ▪ Puede contener billones de nodos y relaciones. ▪ Rápido recorriendo relaciones, este tipo de queries se conoce como transversals ▪ Las escrituras se pueden realizar en cualquier instancia del clúster. ▪ Multilenguaje, proporciona una Api Rest pudiendo utilizarse desde cualquier lenguaje. [29]
	<ul style="list-style-type: none"> ▪ Posee almacenamiento en la nube. ▪ Apoyo de consultas en paralelo. ▪ Precios flexibles y opciones de licencia. ▪ Totalmente transaccional y multi-hilo. [30]
	<ul style="list-style-type: none"> ▪ Recorrido Gráfico y consultas de tipo relacional. ▪ Indexación personalizable. ▪ Gestión de almacenamiento personalizable.[31]



Cuadro 5.4: Cuadro comparativo de gestores de bases de datos orientadas a grafos

Cuadro comparativo de tecnologías	
Nombre	Características
	<ul style="list-style-type: none"> ▪ Es un framework o conjunto de herramientas de software libre para diseño de sitios y aplicaciones web. ▪ Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales. ▪ Es compatible con la mayoría de los navegadores web. ▪ Bootstrap es de código abierto y está disponible en GitHub. ▪ Desde la versión 2.0 también soporta diseños sensibles. Esto significa que el diseño gráfico de la página se ajusta dinámicamente, tomando en cuenta las características del dispositivo usado (Computadoras, tabletas, teléfonos móviles). [32]
	<ul style="list-style-type: none"> ▪ No tiene que agregar clases de responder o lograr cierto estilo. ▪ Muchos prefieren Foundation, ya que ofrece más flexibilidad. ▪ Fácil navegación de su sitio a otro sitio. ▪ Tablas de precios, diseñado para mostrar los precios de un producto a base de suscripción ▪ Las páginas web se ajustan a diferentes dispositivos. [33]


Cuadro 5.5: Cuadro de tecnologías para diseño Web

Cuadro comparativo de tecnologías	
Nombre	Características
	<ul style="list-style-type: none"> ■ Es una herramienta de software para la gestión y construcción de proyectos. ■ Tiene un modelo de configuración de construcción más simple, basado en un formato XML. ■ Utiliza un Project Object Model (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. ■ Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado. ■ El motor incluido en su núcleo puede dinámicamente descargar plugins de un repositorio. ■ Está construido usando una arquitectura basada en plugins que permite que utilice cualquier aplicación controlable a través de la entrada estándar. [34]
	<ul style="list-style-type: none"> ■ Es una herramienta usada en programación para la realización de tareas mecánicas y repetitivas, normalmente durante la fase de compilación y construcción (build). ■ Es un software para procesos de automatización de compilación, similar a Make pero desarrollado en lenguaje Java y requiere la plataforma Java, así que es más apropiado para la construcción de proyectos Java. ■ Esta herramienta, hecha Java, tiene la ventaja de no depender de las órdenes del shell de cada sistema operativo, sino que se basa en archivos de configuración XML ■ Ant utiliza XML para describir el proceso de generación y sus dependencias. [35]

Cuadro 5.6: software para la gestión y construcción de proyectos


Cuadro comparativo de tecnologías	
Nombre	Características
	<ul style="list-style-type: none"> ▪ Es un software de control de versiones y open-source. ▪ Fuerte apoyo al desarrollo no lineal, por ende rapidez en la gestión de ramas y mezclado de diferentes versiones. ▪ Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. ▪ Gestión distribuida, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. ▪ Los repositorios Subversion y svn se pueden usar directamente con git-svn. ▪ Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución. ▪ Todas las versiones previas a un cambio determinado, implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio.[36]
	<ul style="list-style-type: none"> ▪ Es software libre bajo una licencia de tipo Apache/BSD. ▪ Utiliza el concepto de revisión para guardar los cambios producidos en el repositorio. ▪ Se sigue la historia de los archivos y directorios a través de copias y renombrados. ▪ Las modificaciones (incluyendo cambios a varios archivos) son atómicas. ▪ La creación de ramas y etiquetas es una operación más eficiente. ▪ Se envían sólo las diferencias en ambas direcciones. ▪ Maneja eficientemente archivos binarios. ▪ El manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado. [37]

Cuadro 5.7: Software para el control de versiones

Cuadro comparativo de tecnologías	
Nombre	Características
	<ul style="list-style-type: none"> ▪ Es un lenguaje de programación de propósito general, concurrente, orientado a objetos ▪ El API de programación es muy sencilla, flexible y extensible. ▪ Podemos implementar los servlets y JSPs que no son procesos independientes, se ejecutan dentro del mismo proceso que la JVM mejorando notablemente el rendimiento y reduciendo la carga computacional y de memoria requeridas. ▪ Las JSPs son páginas que se compilan dinámicamente de modo que el código que se consigue una ventaja en rendimiento substancial frente a muchos lenguajes interpretados. ▪ La especificación de Servlets y JSPs define un API de programación. ▪ Los requisitos para un servidor ya que se puedan desplegar estos componentes para formar aplicaciones web dinámicas completas. [38]

Cuadro 5.8: Lenguaje de desarrollo y tecnologías relacionadas

Teniendo el conocimiento sobre las tecnologías que se tienen al alcance, la tabla 5.9 se plantea utilizar las siguientes tecnologías para el desarrollo del sistema:

Tecnologías a usar	
Tecnología	Ventajas
Bases de datos Orientadas Grafos	<ul style="list-style-type: none"> ▪ Debido a que están orientadas a mapear relaciones con mayor agilidad, y tienen una estructura flexible ▪ Están orientadas a consultas amplias. ▪ Al ser orientadas a grafos brindan la posibilidad de acceder a los datos a través de algoritmos de recorrido de grafos. ▪ Están diseñadas para soportar grande volúmenes de información.
	<ul style="list-style-type: none"> ▪ A pesar de que es una base de datos NoSQL, acepta transacciones ACID. ▪ Tiene una licencia gratuita open-source. ▪ Soporta diferentes lenguajes, entre ellos Java, Python y Ruby. ▪ Es una herramienta en crecimiento que es altamente escalable. ▪ La experiencia que tienen en el campo.



- El diseño de sitios y aplicaciones web es de licencia libre.
- El diseño de las plantillas es basado en HTML y CSS.
- Sus diseños son responsivos, esto quiere decir que las páginas web se ajustan al dispositivo donde se este visualizando
- Es compatible con la mayoría de navegadores.
- Las plantillas son de gran utilidad al trabajar con estilos
- Es una tecnología conocida por el equipo de trabajo.



- Nos permite hacer programación concurrente.
- Para el uso de servidores , Java es eficiente, desde la aparición de la especificación de Servlets y JSP.
- Las JSP's nos permiten la creación de sitios web dinámicos.
- JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor.
- El Código separado de la lógica del programa.
- Las páginas son compiladas en la primera petición.
- Permite separar la parte dinámica de la estática en las páginas web.
- Es una tecnología conocida por el equipo de trabajo.

Tecnologías a usar	
Tecnología	Ventajas
	<ul style="list-style-type: none"> ▪ La herramienta permite gestionar y construir el proyecto de manera sencilla ▪ Permite controlar las dependencias del proyecto. ▪ Existen plugins que permiten automatizar el trabajo de construcción.
	<ul style="list-style-type: none"> ▪ La herramienta de control de versiones es open-source. ▪ Esta disponible para cualquier sistema operativo. ▪ Integrado con github proporciona herramientas para navegar y visualizar las versiones del proyecto, así como permitir el trabajo colaborativo. ▪ Permite revertir cambios en los archivos cuando un archivo llegue a fallar. ▪ Es una tecnología conocida por el equipo de trabajo.

Cuadro 5.9: Tecnologías usadas

Capítulo 6

Prototipo 1: Obtener y abstraer los datos a utilizar

6.1. Análisis

6.1.1. Objetivo

Crear el módulo de conexión de datos apropiado para el manejo de la información de acuerdo a un modelo de datos propuesto diseñado por el equipo de trabajo.

6.1.2. Características

- El modelo de datos propuesto deberá satisfacer las necesidades para trabajar con diferentes artículos y la interacción con sus usuarios.
- El sistema deberá permitir la conexión a la fuente de datos que corresponda con el modelo propuesto.
- El sistema deberá ser capaz de realizar operaciones sobre los datos de la fuente para el correcto funcionamiento de los procesos lógicos del sistema.

6.1.3. Restricciones

- El sistema se verá limitado por el lenguaje de desarrollo Java.
- Para el caso de estudio, el sistema operará con el gestor de base de datos propio de Neo4j.

6.2. Diseño

6.2.1. Modelo de datos

Para la API, el problema fundamental recae en que las recomendaciones pueden ser utilizadas para diversos casos de estudio, buscando una generalización de las características mínimas requeridas para realizar recomendaciones se plantea un modelo de datos utilizando tres principales entidades: Usuarios, Artículos a recomendar, y Categorías como características que permitan clasificar los diferentes artículos. La información que será guardada y

posteriormente consultada por otros módulos del sistema seguirá un esquema propuesto por el equipo de trabajo, éste recopila la estructura de evaluaciones básicas generadas en prácticamente cualquier tipo de artículos, películas, platillos, libros, entre otros como se puede apreciar en la figura 6.1.

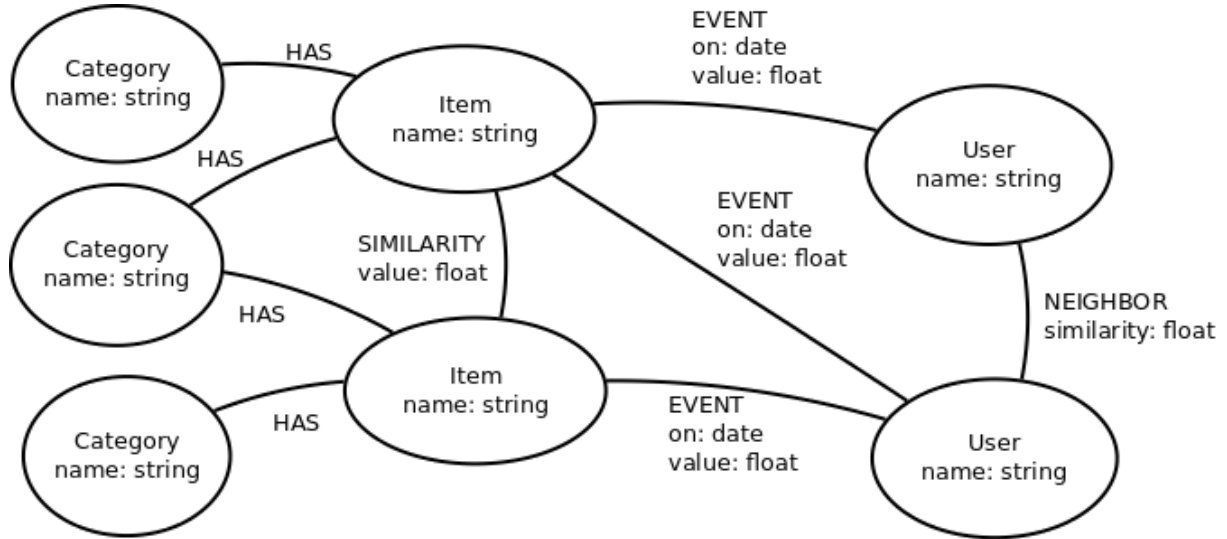


Figura 6.1: Modelo general necesario para el manejo de la información

6.3. Resultados

Para la aplicación cliente correspondiente al caso de estudio de recomendación de platillos, el sistema tendrá un modelo de datos que parte del modelo general mínimo necesario para trabajar la información añadiendo información relevante sobre el caso de estudio que recae en platillos como artículos a recomendar. Considerando al final las siguientes entidades.

- Platillos
- Usuarios
- Restaurantes
- Categorías

La figura 6.2 muestra las entidades utilizadas en el sistema de recomendación de platillos así como las relaciones existentes entre las mismas. En éste los nodos del grafo corresponden a las entidades Usuario, Platillo, Restaurante y Categoría. Donde las relaciones entre ellos se describen de la siguiente manera:

- Un platillo puede ser de una o más categorías.
- Un platillo puede ser servido en uno o más restaurantes
- Un usuario puede agregar un platillo

- Un usuario puede interactuar con un platillo a través de un conteo de clics.
- Un usuario puede evaluar qué tanto le gustó un platillo a través de un rating cuantitativo.

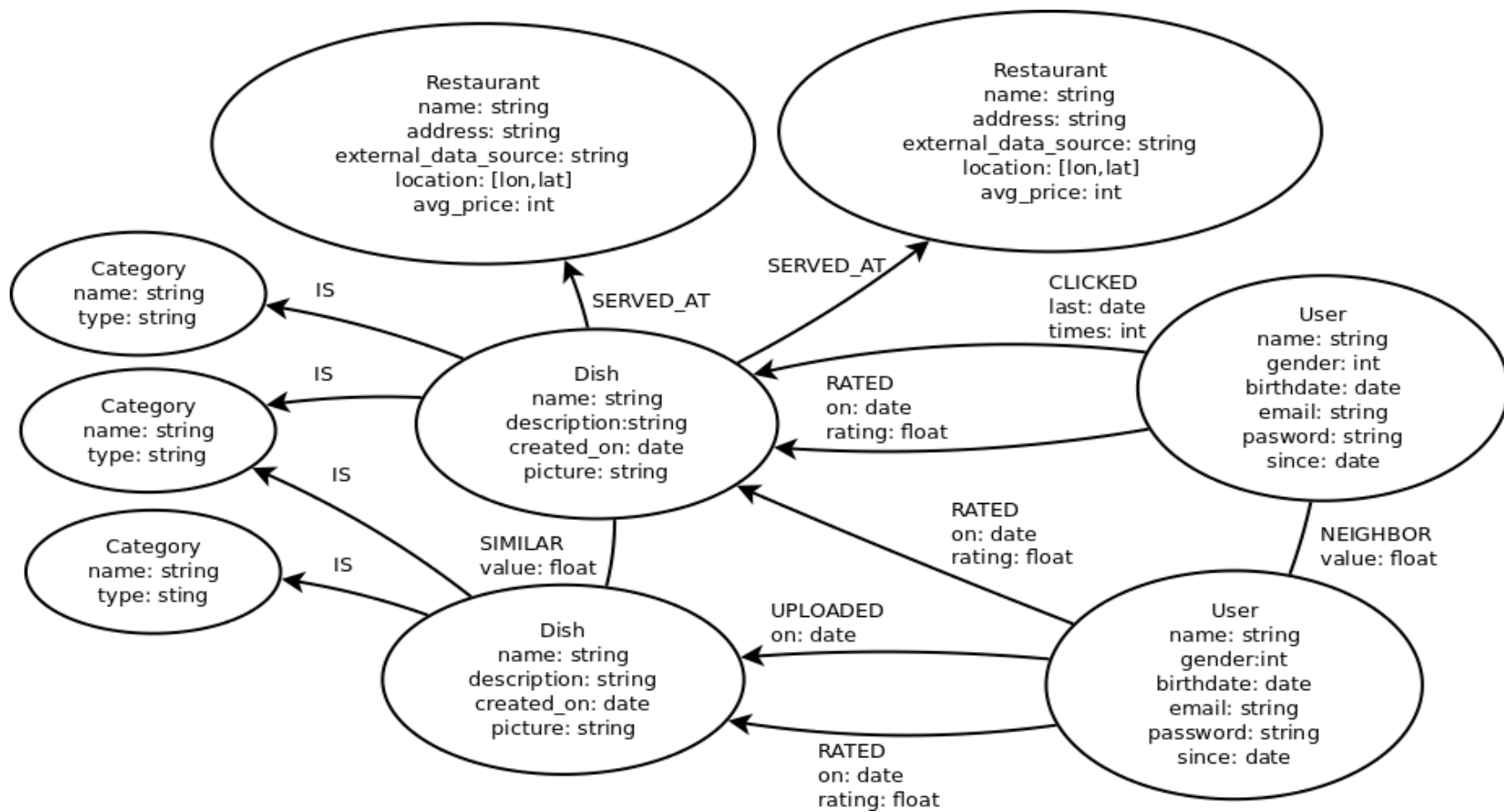


Figura 6.2: Modelo de datos propuesto para el caso de estudio

Utilizando el modelo propuesto por el equipo de trabajo, es posible generar datos de prueba con contenido de platillos así como proponer una serie de categorías que fungirán como características para clasificar los diferentes platillos. Las categorías propuestas se identifican por diferentes rubros como son: tipo de comida, sabor, ocasión, región, salud, temperatura, como podemos notarlas en el anexo 8.1.[23, 24]

Como resultado de la implementación del modelo de datos para el sistema, se obtuvieron datos utilizando la técnica de web-scraping para obtener información de platillos de prueba. La figura 6.3 muestra la siguiente representación gráfica del modelo utilizado en el caso de estudio.



Figura 6.3: Modelo de datos implementado para el caso de estudio

Capítulo 7

Prototipo 2: Visualizar los datos en recomendaciones por contenido

7.1. Análisis

7.1.1. Objetivo

Visualizar recomendaciones bajo el filtrado por contenido de los platillos contenidos en el sistema.

7.1.2. Características

- El sistema permite al usuario visualizar los platillos.
- El sistema muestra recomendaciones de acuerdo a un solo platillo, para mostrar los que tienen una mayor similitud con éste.
- El sistema permite la interacción de usuarios no registrados para obtener recomendaciones personalizadas con base en los platillos que ha visualizado.
- La evaluación de similitud se logra a través de las categorías en común que tengan los platillos entre sí.

7.1.3. Restricciones

- El sistema se ve limitado a la cantidad y características de los platillos existentes actualmente.
- El usuario no registrado no podrá modificar o eliminar los platillos del sistema.

7.2. Diseño

Para la implementación de las recomendaciones por contenido, se consideran las categorías que tienen en común. La cantidad de categorías que poseen en común es identificada por una evaluación de similitud. Dicha evaluación genera un conjunto de relaciones con las que es posible determinar los artículos que se parecen más entre sí y obtener los vecinos más cercanos a cada elemento. Así, para los platillos, las categorías a las que pertenecen,

permiten crear una vecindad para cada nodo donde es posible obtener cuáles son los platillos vecinos que más se parecen entre sí.

7.2.1. Diagrama de clases

Implementando el comportamiento descrito anteriormente se generan las clases mostradas en la figura 7.1 que permiten obtener las recomendaciones por contenido.

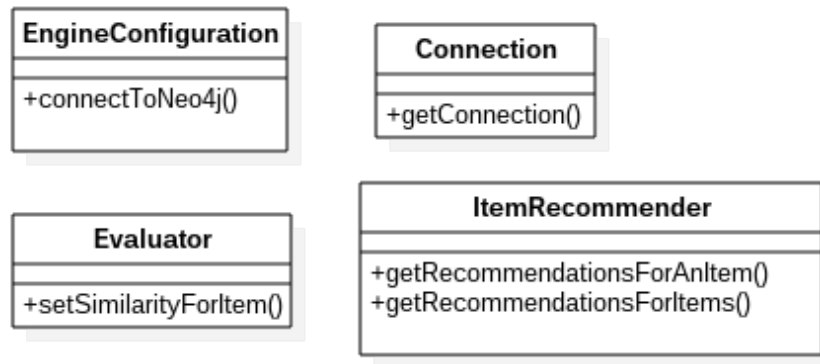


Figura 7.1: Diagrama de clases del prototipo 2

7.2.2. Casos de uso

Integrando la funcionalidad planteada en las clases anteriores dentro de un sistema web que permita la interacción del usuario se plantean los siguientes casos de uso mostrados en la figura 7.2

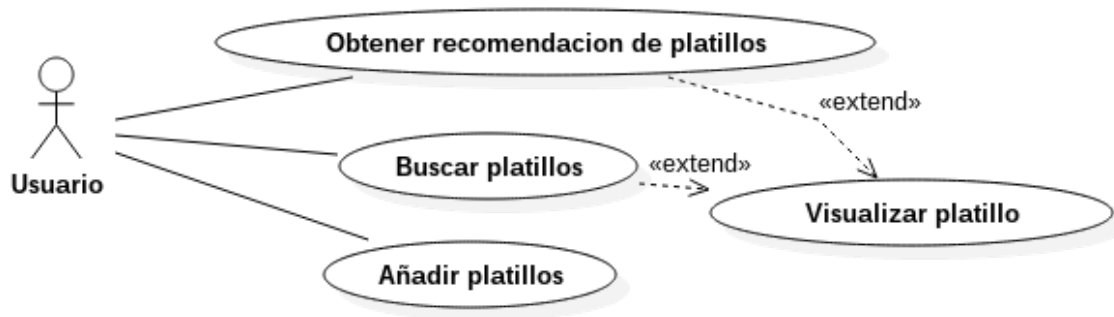


Figura 7.2: Casos de uso del prototipo 2

CU. Obtener recomendación de platillos

Descripción: El usuario obtiene recomendaciones del sistema

Actores: Usuario

Precondiciones: Ninguna

Flujo normal

- El sistema muestra información de platillos relevantes no personalizadas.

Flujo alternativo

- El sistema muestra recomendaciones personalizadas de acuerdo a interacciones previas del usuario.

CU. Buscar platillos

Descripción: El usuario utiliza el sistema de búsqueda para encontrar platillos relacionados.

Actores: Usuario

Precondiciones: Ninguna

Flujo normal

- El usuario escribe su búsqueda en el sistema.
- El sistema obtiene los platillos encontrados, así como platillos relacionados
- El sistema muestra los resultados al usuario

CU. Visualizar platillo

Descripción: El usuario visualiza la información descriptiva de un platillo

Actores: Usuario

Precondiciones: El usuario selecciona un platillo desde el CU. Obtener recomendación de platillos o desde el CU. Buscar platillos

Flujo normal

- El usuario selecciona un platillo de la lista de resultados
- El sistema muestra la información descriptiva de ese platillo.

CU. Agregar platillo

Descripción: El usuario agrega un platillo en el sistema

Actores: Usuario

Precondiciones: Ninguna

Flujo normal

- El sistema muestra el método de entrada para las características de los platillos
- El usuario introduce los parámetros necesarios para registrar el platillo. (Nombre, descripción y categorías a las que pertenece)
- El sistema carga la información del platillo al sistema.

Flujo alternativo

- El sistema comprueba la validez de los datos, si no son correctos, se avisa al actor permitiendo que se corrijan.
- Si el sistema no puede salvar la información del nuevo platillo, se notifica al usuario que el platillo no ha sido agregado.

7.3. Resultados

Tras definir las acciones realizadas en el prototipo, se realiza el desarrollo del sistema Bonappetit, prototipo del sistema final que permite obtener recomendaciones con filtrado por contenido para usuarios no registrados en la plataforma. Esta plataforma se ve implementada en un sistema web cuyo estilo visual es posible de apreciar en la figura 7.3

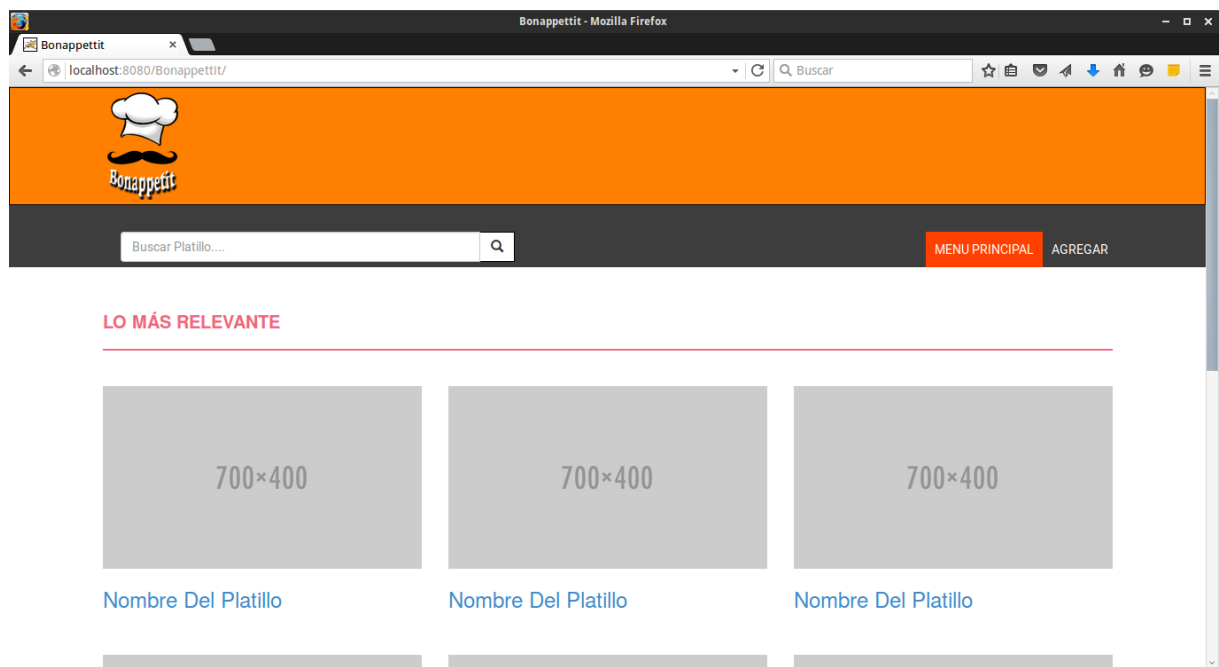


Figura 7.3: Capturas de pantalla del sistema

Así mismo, podemos denotar el resultado de las recomendaciones para un ítem en particular como prueba de la implementación del algoritmo k-NN implementado para reconocer la similitud entre los diversos platillos que se encuentran registrados en el sistema, como se describió en el diseño del prototipo. Debido a la ausencia de usuarios registrados en el sistema, y a la falta de información de restaurantes de una zona en particular, el prototipo se ve limitado a recomendar platillos de acuerdo a sus categorías por filtrado por contenido. Actualmente se cuenta con registro de 843 platillos dentro de la base de datos disponibles para hacer pruebas con las recomendaciones necesarias, se espera este número aumente con el tiempo. Por lo mismo, del modelo de datos propuesto para el caso de estudio en la figura 6.2, para este prototipo se ve limitado al modelo observado en la figura 7.4 solo

permitiendo la recomendación de platillos. Como prueba de su funcionamiento, la figura 7.5 muestra el resultado de la recomendación para la búsqueda de ensaladas, de acuerdo a las características denotadas por las categorías a las que pertenece el primer platillo encontrado.

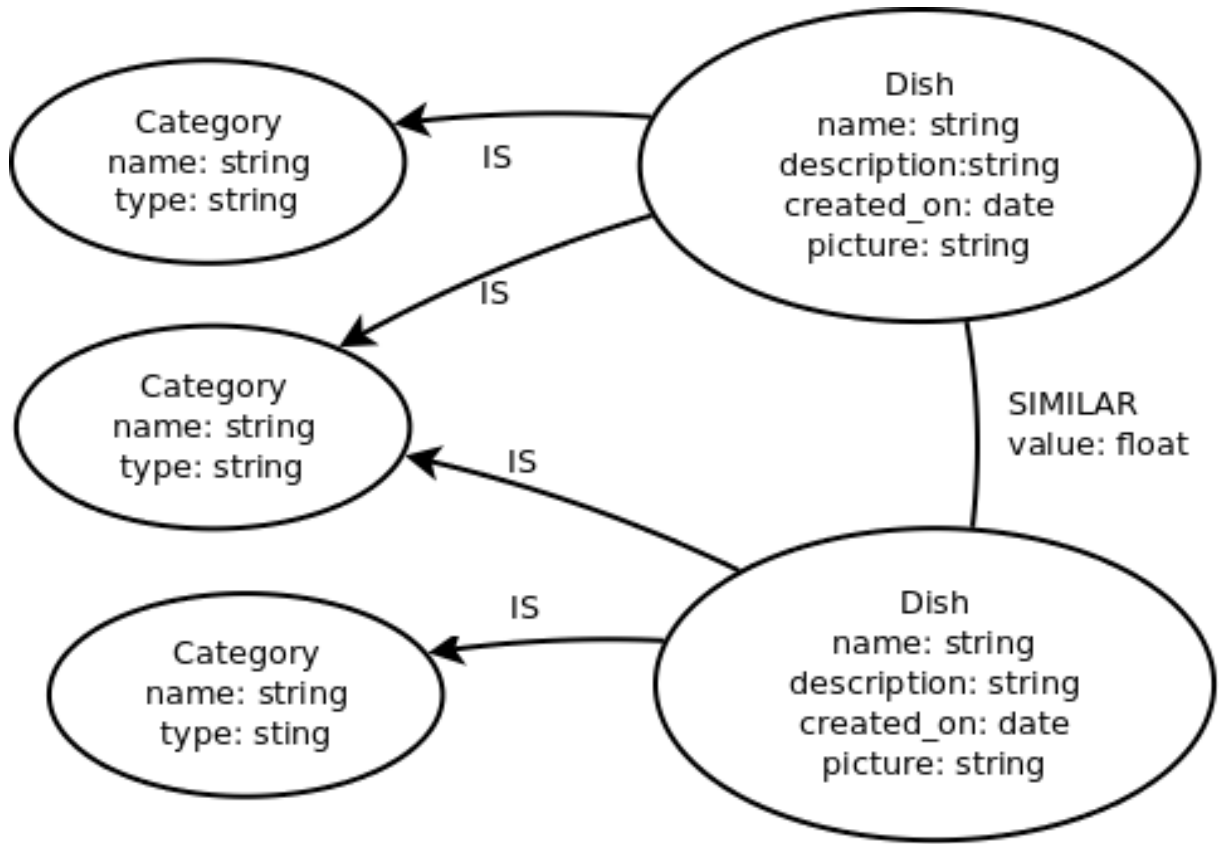
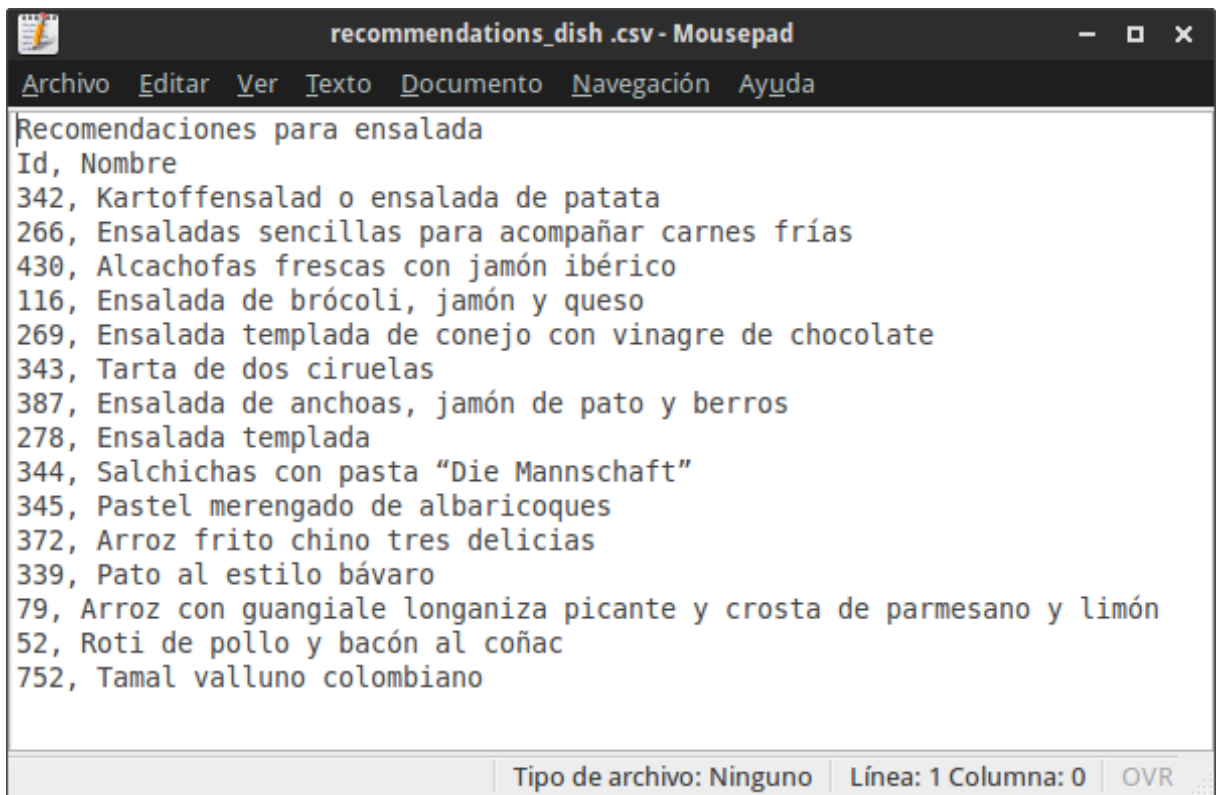


Figura 7.4: Modelo de datos con funcionalidad limitada para el prototipo 2



```
recommendations_dish.csv - Mousepad
Archivo  Editar  Ver  Texto  Documento  Navegación  Ayuda
Recomendaciones para ensalada
Id, Nombre
342, Kartoffensalad o ensalada de patata
266, Ensaladas sencillas para acompañar carnes frías
430, Alcachofas frescas con jamón ibérico
116, Ensalada de brócoli, jamón y queso
269, Ensalada templada de conejo con vinagre de chocolate
343, Tarta de dos ciruelas
387, Ensalada de anchoas, jamón de pato y berros
278, Ensalada templada
344, Salchichas con pasta "Die Mannschaft"
345, Pastel merengado de albaricoques
372, Arroz frito chino tres delicias
339, Pato al estilo bávaro
79, Arroz con guangiale longaniza picante y crosta de parmesano y limón
52, Roti de pollo y bacón al coñac
752, Tamal valluno colombiano
Tipo de archivo: Ninguno  Línea: 1 Columna: 0  OVR
```

Figura 7.5: Recomendación obtenida para la búsqueda de ensaladas

Capítulo 8

Anexos

8.1. Categorías propuestas para el caso de estudio

Id	Tipo	Categoría	Id	Tipo	Categoría
1	Tipo de comida	Panes y masas	48	Ocasión	Merienda
2	Tipo de comida	Pastas	49	Ocasión	Cena
3	Tipo de comida	Bizcochos y galletas	50	Región	Italiana
4	Tipo de comida	Carnes	51	Región	Mediterránea
5	Tipo de comida	Aves	52	Región	Asiática
6	Tipo de comida	Pescados y mariscos	53	Región	Mexicana
7	Tipo de comida	Ensaladas	54	Región	Americana
8	Tipo de comida	Contenido alcohólico	55	Región	Hindú
9	Tipo de comida	Salsas y guarniciones	56	Región	Francesa
10	Tipo de comida	Sopas y cremas	57	Región	Tailandesa
11	Tipo de comida	Arroces	58	Región	Cantonesa
12	Tipo de comida	Legumbres y guisos	59	Región	Japonesa
13	Tipo de comida	Tartas y dulces	60	Región	China
14	Tipo de comida	Helados y sorbetes	61	Región	Medio oriente
15	Tipo de comida	Frutas	62	Región	Alemana
16	Tipo de comida	Verduras	63	Región	Argentina
17	Tipo de comida	Huevos	64	Región	Brasileña
18	Tipo de comida	Lácteos	65	Región	Colombiana
19	Tipo de comida	Frutos secos	66	Región	Coreana
20	Tipo de comida	Encurtidos y conservas	67	Región	Cubana

Id	Tipo	Categoría	Id	Tipo	Categoría
21	Tipo de comida	Postre	68	Región	Española
22	Tipo de comida	Bebida	69	Región	Finlandesa
23	Tipo de comida	Primeros platos	70	Región	Griega
24	Tipo de comida	Segundos platos	71	Región	Holandesa
25	Tipo de comida	Entradas	72	Región	Indonesa
26	Tipo de comida	Sopas y cremas	73	Región	Portuguesa
27	Tipo de comida	Acompañamientos	74	Salud	Bajas en colesterol
28	Tipo de comida	Emparedados	75	Salud	Diabéticos
29	Tipo de comida	Botana	76	Salud	Sin lactosa
30	Tipo de comida	Comida rápida	77	Salud	Celíacos
31	Sabor	Dulce	78	Salud	Alérgicos a los frutos secos
32	Sabor	Salado	79	Salud	Colón irritable
33	Sabor	Ácido	80	Salud	Bajar de peso
34	Sabor	Amargo	81	Salud	Vegetarianos
35	Sabor	Umami	82	Temperatura	Frío
36	Sabor	Picante	83	Temperatura	Templado
37	Ocasión	Halloween	84	Temperatura	Caliente
38	Ocasión	Navidad	85	Gente	Bebes
39	Ocasión	San Valentín	86	Gente	Niños
40	Ocasión	Ocasión especial	87	Gente	Adultos
41	Ocasión	Primavera	88	Gente	Familiares
42	Ocasión	Verano	89	Gente	Adultos mayores
43	Ocasión	Otoño	90	Textura	Líquidas
44	Ocasión	Invierno	91	Textura	Blandas
45	Ocasión	Desayunos	92	Textura	Semi-blandas
46	Ocasión	Almuerzos	93	Textura	Crujientes
47	Ocasión	Comida	94	Textura	Duras

Cuadro 8.1: Categorías propuestas para el caso de estudio

8.2. Diccionario de Datos

Los valores mínimos y máximos de los atributos son determinados por la longitud cuando el tipo de dato amerita una validación por longitud, como lo son las cadenas de texto; y una valoración por rango, para aquellos datos donde solo se aceptan datos que pertenezcan a cierto conjunto, como lo es un rango de numeros, denotado por la forma [inicio:fin].

Entidad	Atributo	Tipo	Min / Max	Descripción
Restaurante	Id	Long	[1:2^63-1]	Es el identificador de restaurante
Restaurante	Name	String	Min: 5, Max: 20	Nombre del restaurante que se registrara en el sistema
Restaurante	Address	String	Min:10, Max:50	Dirección del Restaurante
Restaurante	Longitude	Float	[-180:180]	Indica la longitud de la ubicación geográfica del restaurante
Restaurante	latitude	Float	[-90:90]	Indica la latitud de la posición geográfica del restaurante
Restaurante	averagePrice	Integer	[1:5]	Indica un precio aproximado de los platillos del restaurante, donde 1 identifica a un precio menor y 5 a un precio mayor en el restaurante
Restaurante	External url info	String	Min:20, Max:30	Dirección url de una fuente de datos externa
User	Birthdate	Date	[Fecha actual - 100 años:Fecha actual - 13 años]	Indica la fecha de nacimiento del usuario
User	Email	String	Min:5, Max:50	Email del usuario, necesario para la identificación del mismo
User	Password	String	Min:8, Max:20	Contraseña del usuario. Necesaria para su identificación.
User	since	Date	-	Indicará desde que fecha el usuario se dio de alta en el sistema

Entidad	Atributo	Tipo	Min / Max	Descripción
User	Id	Long	[1:2^63-1]	Identificador numérico del usuario
User	Name	String	Min:5, Max:30	Nombre del usuario
User	Lastname	String	Min:5, Max:30	Apellido del usuario
User	Gender	Char	[F:M]	Género del usuario
Dish	Id	Long	[1:2^63-1]	Identificador numérico del platillo
Dish	Name	String	Min:5, Max:50	Nombre del Platillo
Dish	Description	String	Min:60, Max:500	Contendrá las características del platillo, incluyendo la lista de los ingredientes.
Dish	Picture	String	Min:60, Max:200	Contendrá la ruta de la imagen del platillo
Category	Id	Long	[1:2^63-1]	Contendrá el identificador de la categoría que pertenece
Category	Name	String	Min:5, Max:20	Nombre de la categoría
Category	Type	String	Min:4, Max:15	A que tipo pertenece dicha categoría, ejemplo: Region
Rated	Rating	Float	[0.0:5.0]	ayudara para saber que calificación le dio el usuario al platillo recomendado
Rated	On	Date	-	ayudara a saber en que fecha hizo ese rating
Cliked	Last	Date	-	La ultima fecha en que visualizo el platillo
Cliked	Times	Int	[1:2^31-1]	Las veces en las que a visualizado ese platillo
Uploaded	On	Date	-	En que fecha el usuario subió el platillo

Cuadro 8.2: Diccionario de datos

Glosario

- **API:** Application Programming Interface, conjunto de rutinas, protocolos o herramientas para construcción de software.
- **ACID:** Es un acrónimo de Atomicity, Consistency, Isolation and Durability: Atomicidad, Consistencia, Aislamiento y Durabilidad en español.
- **Atomicidad:** Si una operación consiste en una serie de pasos, todos ellos ocurren o ninguno, es decir, las transacciones son completas.
- **Atributo:** Los atributos son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. Los atributos se guardan en variables denominadas de instancia, y cada objeto particular puede tener valores distintos para estas variables.
- **Aislamiento:** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error. Esta propiedad define cómo y cuándo los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes.
- **Base De Datos:** Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
- **Caso De Uso:** Es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso.
- **Consistencia:** Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de Integridad de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido.
- **Durabilidad:** Persistencia. Es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema y que de esta forma los datos sobrevivan de alguna manera.
- **Entidad:** Representa una cosa u objeto del mundo real con existencia independiente, es decir, se diferencia únicamente de otro objeto o cosa, incluso siendo del mismo tipo, o una misma entidad.
- **Framework:** Es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.
- **Grafo:** Estructura la información en forma de nodos y relaciones.
- **Prototipo:** Es una representación de un sistema, aunque no es un sistema completo, posee las características del sistema final o parte de ellas.
- **Web Scrapping:** Técnica implementada mediante programas de software para extraer información de sitios web.

Bibliografía

- [1] Roger Loaiza. *¿Qué es la inteligencia artificial?. “De la información a la informática”*. Obtenido desde http://bvs.sld.cu/revistas/san/vol2_2_98/san15298.htm el 1 de mayo de 2015.
- [2] Enrique Herrera-Viedma, *Sistemas de recomendaciones: herramientas para el filtrado de información en Internet*, 2004. Obtenido desde <http://www.upf.edu/hipertextnet/numero-2/recomendacion.html> el 2 de mayo de 2015.
- [3] Marcos Merino. *¿Qué es una API y para qué sirve?* Obtenido desde: <http://www.ticbeat.com/tecnologias/que-es-una-api-para-que-sirve/> el 4 de mayo de 2015.
- [4] Paul Resnick and Hal R. Varian, *Recommender Systems*. 1997. Obtenido desde: https://www.ischool.utexas.edu/~i385d/readings/Resnick_Recommender_97.pdf
- [5] Robin Burke, California State University, Fullerton *Hybrid Recommender Systems: Survey and Experiments*. Obtenido desde: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.8200&rep=rep1&type=pdf>
- [6] Gediminas Adomavicius, Alexander Tuzhilin. *Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions*. Obtenido desde: <http://homepages.dcc.ufmg.br/~nivio/cursos/ri13/sources/recommender-systems-survey-2005.pdf>
- [7] Chad Vicknair, Michael Macias, Zhendong Zhao, Xiaofei Nan, Yixin Chen, Dawn Wilkins. *A Comparison of a Graph Database and a Relational Database*. Obtenido desde: http://cs.olemiss.edu/~ychen/publications/conference/vicknair_acmse10.pdf
- [8] Renzo Angles , Claudio Gutierrez. *Survey of Graph Database Models* Obtenido desde: http://www.dcc.uchile.cl/TR/2005/TR_DCC-2005-010.pdf
- [9] Bases de Datos Orientadas a Grafos. Obtenido desde <http://www.bigdatahispano.org/noticias/bases-de-datos-orientadas-a-grafos/>
- [10] Brian Proffitt, *What APIs Are And Why They’re Important*. Sep 19, 2013. Obtenido desde: <http://readwrite.com/2013/09/19/api-defined>
- [11] Sofía Marina Pepa, *Suite de algoritmos de recomendación en aplicaciones reales*. Obtenido desde: https://repositorio.uam.es/bitstream/handle/10486/660903/marina_pepa_sofia_tfg.pdf?sequence=1
- [12] Spotify. Obtenido desde: <https://www.spotify.com/mx/about-us/contact/>

- [13] Sistema Generador de Recomendaciones para una Tienda En-Línea de Videojuegos, 2010. ESCOM IPN.
- [14] Gregory Piatetsky. Prediction.io open source machine learning server, 10 de abril de 2014. Obtenido desde: <http://www.kdnuggets.com/2014/04/prediction-io-open-source-machine-learning-server.html> el 6 de mayo de 2015.
- [15] Easyrec, Obtenido desde <http://easyrec.org/> el 6 de mayo de 2015
- [16] REST API. Obtenido desde: http://easyrec.sourceforge.net/wiki/index.php?title=REST_API_v0.98 el 6 de mayo de 2015.
- [17] Netflix. Obtenido desde: <https://help.netflix.com/es/node/412> 6 de mayo de 2015.
- [18] Lenskit: Open-Source Tools for Recommender Systems. Obtenido desde <http://lenskit.org/>
- [19] Neo4j Obtenido desde <http://xurxodeveloper.blogspot.mx/2014/03/neo4j-una-base-de-datos-nosql-orientada.html/>
- [20] Cómo Funciona Neo4j Obtenido desde <http://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos/>
- [21] Neo4j-Rendimiento Obtenido desde <http://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos/>
- [22] Oracle y Java, características. Obtenido desde <http://www.oracle.com/es/technologies/java/features/index.html>
- [23] Alergia a los alimentos. Obtenido desde: <http://www.laaleria.com/tipos-alergia/alimentos/>
- [24] ReceTags: Recetario en línea. Obtenido desde: <http://www.recetags.com/>
- [25] Tipos de API. Obtenido desde http://www-01.ibm.com/support/knowledgecenter/SS6PEW_9.4.0/com.ibm.help.custom.apis.doc/API_API_Types.html?lang=es
- [26] Bases de datos relacionales. Obtenido desde J. J. King, «QUIST: A System for Semantic Query Optimization in Relational Data Bases», Proc. of the International Conf. on Very Large Databases (1981), páginas 510-517
- [27] Bases de datos no relacionales. Obtenido desde `\T1\guillemotleftNoSQLRelationalDatabaseManagementSystem:HomePage\T1\guillemotright.Strozzi.it.2deoctubrede2007.Consultadoel10deNoviembrede2015.`
- [28] Bases de datos orientadas a grafos. Obtenido desde <http://graphbase.net/>
- [29] Bases de datos orientadas a grafos. Obtenido desde <http://xurxodeveloper.blogspot.mx/2014/03/neo4j-una-base-de-datos-nosql-orientada.html>

- [30] InfiniteGraph. Obtenido desde Joyce Wells (June 26, 2013). "DBTA 100: The Companies That Matter Most in Data". Database Trends and Applications. Retrieved September 8, 2014.
- [31] Infogrid. Obtenido desde <http://infogrid.org/trac/>
- [32] Bootstrap. Obtenido desde <http://getbootstrap.com/2.3.2/>
- [33] Foundation. Obtenido desde <http://foundation.zurb.com/>
- [34] Maven. Obtenido desde <http://chuwiki.chuidiang.org/index.php?title=Categor%C3%ADa:Maven>
- [35] Ant. Obtenido desde <http://ant.apache.org/>
- [36] Git. Obtenido desde <https://git-scm.com/>
- [37] Subversion <http://svnbook.red-bean.com/nightly/es/svn-ch-1-sect-1.html>
- [38] Java <http://www.oracle.com/technetwork/java/index.html>