Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: *Plugin* para el cálculo de los k vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV.

Autores:

Randy Ruvira Castillo.

Karell González Caro.

Tutor:

Ing. Miguel Morciego Varona.

La Habana, 2015

"Año 57 de la Revolución"

Declaración de autoría

Declaramos ser autores del presente trabajo de diploma y concedemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____.

_____ _____

Autor Autor

Randy Ruvira Castillo Karell González Caro

_____

Tutor

Ing. Miguel Morciego Varona

“Lo importante en ciencia no es tanto obtener nuevos hechos como descubrir nuevas formas de pensar sobre ellos.”

William Lawrence Bragg

Datos de Contacto

Tutor: Ing. Miguel Morciego Varona.

Categoría Científica: Ingeniero.

Correo Electrónico: mmorciego@uci.cu

Síntesis del Tutor: Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas en el año 2014.

Agradecimientos

Karell:

A mi mamá y a mi papá, por ser los artífices principales de este sueño hecho realidad. Por su dedicación y entrega. Por haberme dedicado la vida entera y nuca dejar que me rindiera en los momentos difíciles en que pensé dejar la universidad. Los amo y son mi razón de ser.

A mi queridísima abuelita Carmela, que domingo tras domingo reza por mí en la iglesia y le pide a Dios por mi salud y mi bienestar. Hoy se ve el resultado de tus plegarias abuelita. Gracias por todo el amor que me has entregado desde que nací.

A todos mis tías y tíos, paternos y maternos, por hacerme sentir una persona especial, muy querida y tenerme siempre en lo más alto de la familia. Gracias a todos por su cariño.

A mis tíos Julián y Marylin por la responsabilidad que han asumido conmigo y con mi mamá. Gracias por existir y por todos los consejos que día a día me dan. Les estoy muy agradecido.

A todos mis primos por estar siempre pendientes de mí. Por ser tan apegados a mí y demostrarme que siempre voy a poder contar con ellos. A mi primo Geyser por ser súper atento conmigo y estar siempre ahí cuando lo necesito.

A nuestro tutor Miguel Morciego Varona, "el Genius", por toda la ayuda que nos dio y todo el tiempo que nos dedicó aun cuando no era el tutor de nosotros.

Por supuesto que en mis agradecimientos no puede faltar mi dúo de tesis. Quién más, el otro integrante del dúo más completo que ha tenido jamás la UCI. Los tan conocidos "Karell y Randy". Gracias Randy por tu amistad y tus consejos. Más que mi amigo fuiste mi hermano en la universidad.

A mis amigos de siempre, mi gente del Politécnico que hoy están aquí presentes demostrándome que más que amigos siempre seguimos siendo una familia aún cuando las carreras nos separaron.

A mis grandes amigos de la universidad por toda su dedicación y el esfuerzo que siempre hicieron por mí. Por las largas horas de estudio que me dedicaron.

A todos los socios fuertes con los que compartí aula y residencia, incluso a los que no son de mi año. A todos gracias por hacerme la estancia en la universidad mucho más amena.

A todos los socios del fútbol, de la escuela y del barrio, por todas las horas que pasamos distrayéndonos.

A todas mis amistades que por una razón u otra no pudieron culminar la universidad.

En fin quiero agradecer a todas aquellas personas que de una forma u otra hicieron posible que me convirtiera en Ingeniero, y que muchos no están aquí hoy.

Agradecimientos

Randy:

A mi mamá y a mi papá, por ser los padres que cualquier hijo desearía tener. A mi mamá por su cariño y su comprensión en todo momento de mi carrera y de mi vida, porque sin saber mucho de una universidad yo siento que ella también se gradúa este día porque la sentía al lado mío en todos los momentos por los que pase, a mi papá por siempre crearme el espíritu de superación y las ambiciones de adquirir conocimiento, porque siempre estuvo ahí cuando lo necesite para cualquier cosa, porque tu desmientes eso que dicen que "padre es cualquiera" padre no es cualquiera padre eres tu papi.

A mi hermano por enseñarme a su manera las cosas de la vida, por ayudarme a no cometer los errores que el cometió, por demostrarme que la vida no es tan sencilla como yo creía, y por darme el regalo del ser tío de una niña maravillosa.

A esa niña llamada Patricia que más que mi sobrina es mi hija, porque cada día que pasa se parece más a mí y eso me hace el tío más orgulloso del mundo, para que dentro de unos años cuando leas esto sepas que tú no eres mi sobrina sino que también eres mi niña.

A mi novia por aguantarme estos cinco años, por ayudarme en mi transcurso de niño a hombre, por ser mi compañera, mi amiga y en algunos casos mi enemiga. Por enseñarme lo importante que es el cariño y por enseñarme que el amor si existe solo que no todo el mundo lo ve. Gracias mi amor por estar conmigo estos cinco años.

A mis dos abuelas Mima y Guille por estar siempre pendientes de mí.

A mi cuñada Yanaris y Dania, la primera por ser como una hermana mayor para mí y aconsejarme cuando lo necesitaba, y además también gracias regalarme la bendición de ser tío. La segunda por ser mi cuñada y mi amiga, por aconsejarme tanto cuando empecé con tu hermana y por regalarme otro amigo que es Becker al cual también le agradezco por los momentos que pasamos juntos.

A mis suegros por acogerme como otro hijo, a mi suegra por malcriarme y educarme a la misma vez, a mi suegro por acogerme como su tercer hijo varón, enseñarme cosas de cocina, construcción y de la vida.

A Teresita por cuidar siempre de mi papá en todos los momentos y por enseñarme varias cosas de la vida.

A toda la familia de mi novia, especialmente para su hermano Duviel y su familia (incluida Chucha y Escuela), a sus tías, primas, primos y abuelos, a todos gracias por acogerme como uno más de la familia.

A nuestro tutor Miguel, por guiarnos y sacrificarse con nosotros a pesar de no ser nuestro tutor desde el principio.

A esos amigos y hermanos que me ha dado la UCI. Especialmente a mi dúo de tesis Karell por estar siempre conmigo en todas las batallas imposibles, por regalarme tu amistad todos estos años, porque de verdad eres más que un amigo para mí, eres mi hermano.

A la mama de Karell por todos sus consejos y criterios sobre la tesis.

A todas aquellas personas que hicieron posible que hoy pueda decir "Soy Ingeniero en Ciencias Informáticas".

Dedicatoria

Karell:

A mi mamá y a mi papá que siempre soñaron con que llegara el día en que yo me hiciera Ingeniero, y para que se cumpliera hicieron hasta lo imposible. Hoy comienzo con esta tesis a devolverle toda la vida que me han dedicado. Gracias por ser los mejores padres del mundo, por todo el cariño que me han entregado, y siempre haber confiado en mí. Quiero particularmente dedicarle la tesis a mi papá que por circunstancias de la vida hoy no está aquí físicamente conmigo, pero siempre me ha dado su apoyo y sé que le hubiera encantado poder estar compartiendo conmigo en este día tan especial. En estos momentos está esperando mi llamada ansioso por que le diga que su hijo ya es Ingeniero. Siempre te llevo muy presente papi.

Por último quiero terminar estas palabras dedicándole la tesis a una persona muy especial en mi vida. A "Nené" que hoy no está presente aquí en este salón por situaciones de la vida, pero yo sé lo que significo para ella. Te dedico esta tesis por todo el amor y la atención que siempre me diste. Por todo el apoyo y la ayuda que me brindaste cada vez que me hizo falta. Por todos los días que pasamos juntos. Por los buenos y no tan buenos momentos que compartimos. Por todo el tiempo que estuviste a mi lado. ***Que hubiese sido de mí sin ti…***

Dedicatoria

Randy:

A mi mamá: Por tener siempre tus brazos abiertos y tu corazón lleno de amor hacia mí, por creer siempre en mí desde el principio de esta travesía, y por ser lo más grande que tengo en este mundo.

A mi papá: Por cuidarme siempre, por ser mi guía, mi horizonte, por estar ahí cuando te necesitaba, y sobre todo, por darme la oportunidad de ser tu hijo

A mi hermano: Por lo que representas para mí que no es solo es un hermano, sino también un amigo y un padre. Por ser ese hermano del cual cualquiera estaría orgulloso.

A mi novia: Porque eres una de las mejores cosas que me ha pasado en la vida y para que esto te de fuerzas para que el año que viene tu estés aquí donde estoy yo hoy.

Resumen

Con el objetivo de brindarle a un usuario de la web la información que realmente le pudiera interesar, sin necesidad de una búsqueda trabajosa, surgen los sistemas de recomendación. La plataforma AGORAV es un sistema que se encarga de gestionar y publicar grandes volúmenes de archivos multimedia, contando con un sistema de recomendación que utiliza la técnica de filtrado colaborativo para generar las recomendaciones. Este sistema presenta como inconveniente, la utilización de una cantidad fija de usuarios con los cuales calcula la similitud respecto al usuario sobre el cual se desea generar una recomendación. La presente investigación surge a partir de la necesidad de encontrar las recomendaciones idóneas para un usuario, logrando un balance entre la precisión y el tiempo de respuesta de lo recomendado. El *plugin* desarrollado arroja una cantidad variable de usuarios con los cuales se busca la similitud respecto al actual, con el fin de generar una predicción sobre sus posibles preferencias. Además brinda la posibilidad de que se pueda desplegar en cualquier entorno logrando un balance entre la precisión y el tiempo de respuesta de las recomendaciones. La solución fue desarrollada haciendo uso del lenguaje C++ y la técnica del Agregado de Vecinos seleccionada de las diferentes soluciones existentes dentro de los Algoritmos para el Cálculo de la Vecindad.

Palabras claves: agregado de vecinos, filtrado colaborativo, precisión, sistema de recomendación.

Abstract

With the aim of providing to a web user the information that could be really interesting, without a laborious search, arise recommendation systems. The AGORAV platform is a system that manages and publishes large volumes of multimedia files, with a recommendation system that uses collaborative filtering technique to generate recommendations. This system has the drawback, the use of a fixed number of users for which the similarity is calculated regarding the user which is to generate a recommendation. This research arises from the need to find the best recommendations for a user, achieving a balance between accuracy and response time about it was recommended. The plugin developed casts a variable number of users with which the similarity is sought from the current, in order to generate a prediction about the possible preferences. It also offers the possibility that it can be deployed in any environment achieving a balance between accuracy and time response of the recommendations. The solution was developed using the C ++ language and the technique selected was the adding neighbors from the different solutions within the existing algorithms for the calculation of the neighborhood neighbors.

Keywords: *adding neighbors, collaborative filtering, precision, recommendation system.*

Índice de Contenido

Índice de Tablas

Índice de Figuras

Introducción

El uso de Internet se ha difundido ampliamente en los últimos tiempos, convirtiéndose en la mayor fuente de información digital. Este avance ha traído consigo el desarrollo de la tecnología mundial y el medio idóneo para publicar y obtener información. En la actualidad todo ha cambiado respecto a la información, de la escasez se ha pasado a la saturación. Inicialmente en la década de los setenta, cuando se realizaba una búsqueda de información, aparecían diferentes artículos, hoy en día aparecen el doble (Nieto, 2007), por lo que es más complicado la selección de los mismos. El problema actual de los usuarios de la web, radica en cómo seleccionar lo que les interesa a partir de los resultados encontrados en sus búsquedas. A raíz de esta problemática comienzan a tomar auge los sistemas de recomendación a principios de la década de los 90 del pasado siglo, cuando se realiza dentro de los servicios de grupos de noticias (*newsgroups*) el servicio de filtrado de noticias. Este servicio le permitía a su comunidad de usuarios acceder exclusivamente a aquellas noticias que podían ser de su interés (Peis, y otros, 2008).

El primer sistema de recomendación que surgió fue el llamado Tapestry, desarrollado por XeroX PARC. Tapestry es un sistema que permite almacenar el *feedback*[1] de los usuarios sobre los artículos o noticias que éstos han leído y posteriormente ser utilizado por otros usuarios que aún no han leído el artículo o noticia, para establecer si la información del documento es relevante o no. En un principio este tipo de sistemas fue adoptado con el nombre de filtro colaborativo (*collaborative filter*) dado que permite que los usuarios creen filtros a través de sus ítems de interés (en el caso de Tapestry, artículos o noticias), y colaborativo pues los usuarios añaden las anotaciones con las opiniones sobre los documentos, colaborando así de esta manera a la mejora de los resultados esperados. La idea que subyace tras los sistemas de recomendación por Filtrado Colaborativo es encontrar usuarios con gustos similares a los de un usuario determinado y a partir de ahí poder recomendarle a este último, información que desconoce pero que se sabe que es del gusto de aquellos con los que tiene similitud (Nieto, 2007).

Cuba no se encuentra ajena a esta tendencia de la recomendación de audiovisuales, es por ello que la Universidad de las Ciencias Informáticas prevé el desarrollo de varios sistemas recomendadores, para facilitar el acceso a la información por parte de los usuarios de la web, ayudando a separar lo que pudiera ser interesante para ellos. Particularmente el Centro de Geoinformática y Señales Digitales de la Facultad

[1] Es la reacción, respuesta u opinión que nos da un interlocutor como retorno sobre un asunto determinado (7Graus, 2015).

6 desarrolla como uno de sus productos, una plataforma web para la transmisión de videos, la cual ha sido implementada en la propia universidad y es del interés de varios clientes externos.

La plataforma AGORAV presenta una arquitectura modular lo que permite el desarrollo creciente de funcionalidades. Integrada a ella se encuentran varios *plugins* de recomendación, los cuales utilizan diferentes algoritmos y criterios para su implementación. Entre ellos se encuentra el *plugin* de recomendación por filtrado colaborativo, el cual utiliza una cantidad fija de colaboradores, y las variables que afectan directamente al sistema de recomendación son muy cambiantes y en ocasiones ambiguas en los distintos entornos a los que está destinada la plataforma AGORAV. Esto trae consigo que la calidad de las recomendaciones y su tiempo de respuesta no se adapten al ambiente en el que se va a utilizar.

A partir de las consideraciones anteriormente expuestas se deriva como **problema de la investigación**: ¿Cómo lograr un balance entre la precisión y el tiempo de respuesta de las recomendaciones por filtrado colaborativo? Para darle solución al problema en la presente investigación se define como **objeto de estudio**: algoritmos de filtrado colaborativo. Se identifica como **objetivo general:** Desarrollar un *plugin* adaptable a distintos entornos que logre un balance entre la precisión y el tiempo de respuesta de las recomendaciones que utilizan la técnica de filtrado colaborativo. Se delimita como **campo de acción:** algoritmos de filtrado colaborativo para el cálculo de los k vecinos más cercanos.

A continuación se definen las siguientes preguntas científicas con el objetivo de desglosar el problema de la investigación:

1. ¿Cuáles son los algoritmos que se utilizan en los sistemas de recomendación por filtrado colaborativo?
2. ¿Cuáles son los algoritmos para la búsqueda de la vecindad?
3. ¿Qué algoritmos para la búsqueda de la vecindad se utilizan para lograr un balance entre la precisión y el tiempo de respuesta de las recomendaciones?
4. ¿Cómo lograr un balance entre la precisión y el tiempo de respuesta de las recomendaciones?

Para lograr el objetivo general se tendrán en cuenta las siguientes tareas de la investigación:

1. Establecimiento de los fundamentos teórico-metodológicos para el desarrollo de algoritmos de filtrado colaborativo.

2. Caracterizar los algoritmos de filtrado colaborativo en lo relativo a la precisión y el tiempo de respuesta de las recomendaciones.
3. Desarrollo del sistema informático para el cálculo de los k-vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV.
4. Validar la contribución lograda a través de la introducción del sistema informático en los grados de precisión y tiempo de respuesta de las recomendaciones por filtrado colaborativo, para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV.

Para abordar la realidad conceptual y práctica de los algoritmos de filtrado colaborativo, en el transcurso de la investigación se utilizaron varios **métodos científicos**, clasificados en teóricos y empíricos.

Entre los **métodos teóricos** empleados se encuentran:

Analítico-Sintético: Se emplea para analizar los diferentes algoritmos de recomendación por filtrado colaborativo, con el propósito de establecer un balance entre la precisión y el tiempo de respuesta de las recomendaciones orientadas a distintos entornos.

Histórico-Lógico: Permite realizar un análisis profundo sobre el estado actual y la evolución que han tenido los algoritmos de filtrado colaborativo, mostrando los resultados realizados por la comunidad científica durante los últimos cinco años.

Modelación: Ofrece parte de la información necesaria acerca del objeto que se estudia, se trata de explicar la realidad con la creación de diagramas, los cuales pueden ser presentados en sustitución de la realidad. Se usó en el modelo de dominio para comprender el entorno donde se desarrollará el algoritmo.

Entre los **métodos empíricos** empleados se encuentran:

Entrevista: Este método se utilizó para identificar los requisitos y funcionalidades que debe cumplir el *plugin* a implementar, aplicándosele al líder del proyecto CPM (Catalogación y Publicación de Medias) y a uno de sus desarrolladores.

El presente trabajo de diploma está conformado por 3 capítulos:

Capítulo 1 Fundamentación Teórica sobre los algoritmos para la selección de la vecindad: En este capítulo se definen los conceptos asociados a la problemática identificada. Es mostrada la situación actual,

se definen la metodología de desarrollo, el lenguaje de modelado unificado, además de las herramientas y tecnologías a utilizar.

Capítulo 2 Análisis y diseño del componente para el cálculo de los k vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV: Se especifican los requisitos funcionales y no funcionales que debe cumplir el *plugin* a desarrollar. Se seleccionan la arquitectura y los patrones de diseño, así como la descripción de los casos de uso, los artefactos y diagramas resultantes del diseño del *plugin*.

Capítulo 3 Implementación y pruebas del componente para el cálculo de los k vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV: Se abordan aspectos referentes a la implementación del *plugin*, como son los diagramas de despliegue y de componentes. Además se exponen los resultados que arrojaron las pruebas realizadas al *plugin* desarrollado, como parte de un proceso para verificar el algoritmo implementado como solución al problema y comprobar su correcto funcionamiento.

Capítulo 1: Fundamentación Teórica sobre los algoritmos para la selección de la vecindad

En este capítulo se realiza un análisis de la problemática actual que presenta la técnica de filtrado colaborativo que se utiliza en la plataforma AGORAV. Se definen los conceptos que están estrechamente vinculados al objeto de estudio, además de realizar un estudio de las herramientas y tecnologías que servirán para la elaboración del *plugin*. También se especifica la metodología de desarrollo de software a utilizar, mencionando sus principales características.

1.1 Conceptos asociados al dominio del problema

A continuación se relacionan los principales elementos que están asociados al desarrollo de la investigación. Para ello, se esclarecen algunos conceptos importantes para el entendimiento y el dominio del problema.

1.1.1 Sistema de Recomendación

Estos sistemas se encargan de suministrar a los usuarios que interactúen con el sistema, información personalizada y diferenciada sobre determinados productos y/o servicios que pueden ser de interés para ellos, modificando el proceso de navegación y búsqueda, lo que constituye una ventaja para los clientes, que encontrarán lo que necesitan de una forma más rápida, cómoda y fácil dentro de las enormes bases de datos que ofertan las tiendas electrónicas en Internet. Además descubrirán nuevos elementos que le puedan ser atractivos haciendo uso de los sistemas de recomendación, que de no existir, les hubiera sido mucho más difícil o incluso imposible de encontrar (Rodríguez, 2008). Los sistemas de recomendación se clasifican atendiendo a su funcionamiento, dando lugar a varios tipos de sistemas de recomendación:

- Sistemas de Recomendación Basados en Contenido: Un sistema de recomendación basado en contenido es aquel en el cual las recomendaciones son realizadas basándose en un perfil creado a partir del análisis del contenido de los objetivos que el mismo usuario ha comprado, utilizado o visitado en el pasado (Balabanovic, 1997).
- Sistemas de Recomendación Basados en Conocimiento: Los sistemas de recomendación basados en conocimiento realizan inferencia entre las necesidades y preferencias de cada usuario para sugerir recomendaciones (Guo, 2006). A diferencia de otros sistemas de recomendación, los basados en conocimiento no dependen de grandes cantidades de información sobre objetos

puntuales (basados en contenido) y usuarios particulares (colaborativos) sino que lo único que necesitan es tener un conocimiento general sobre el conjunto de objetos y un conocimiento informal de las necesidades del usuario.

- Sistemas de Recomendación Basados en Utilidad: Los recomendadores basados en utilidad recomiendan utilizando el cálculo de la utilidad de cada uno de los servicios para el usuario. Evidentemente, el problema clave a resolver aquí es cómo crear una función que defina la utilidad para cada usuario y que después pueda ser empleada de manera adecuada para la recomendación (Burke, 2002).
- Sistemas de Recomendación Colaborativos: Los sistemas de recomendación basados en un filtrado colaborativo son aquellos en los que las recomendaciones se realizan basándose solamente en los términos de similitud entre los usuarios (Balabanovic, 1997). Es decir, los sistemas colaborativos recomiendan objetos que son del gusto de otros usuarios de intereses similares.

1.1.2 Filtrado Colaborativo

El filtrado colaborativo es una técnica utilizada por algunos sistemas recomendadores para tratar de predecir lo que realmente interesa a un usuario. En general, el filtrado colaborativo es el proceso de filtrado de información o modelos, que usa técnicas que implican la colaboración entre múltiples agentes y fuentes de datos. Los métodos de filtrado colaborativo se han aplicado a muchos tipos de información, incluyendo la detección y control de datos (como en la exploración mineral, sensores ambientales en áreas grandes o sensores múltiples y datos financieros) tales como instituciones de servicios monetarios que integran diversas fuentes bancarias, o en formato de comercio electrónico y aplicaciones web 2.0 donde el foco está en la información del usuario. Esta discusión se centra en el filtrado colaborativo para datos de usuario, aunque algunos de los métodos y enfoques pueden aplicarse a otras aplicaciones. Dentro del flujo de los sistemas de recomendación uno de los procesos principales es conformar la vecindad, que no es más que los usuarios más parecidos al usuario actual y a partir de estos se generan las recomendaciones. Los sistemas de recomendación por filtrado colaborativo utilizan diferentes algoritmos a la hora de generar las recomendaciones, entre los que se encuentran (Nieto, 2007):

- Algoritmos basados en vecinos más cercanos.
- Algoritmos basados en elementos.
- Predictores *Slope-One*.

En el caso de los algoritmos basados en vecinos más cercanos, cabe señalar que fueron los primeros algoritmos de filtrado colaborativo en implementarse. En primer lugar es necesario medir los parecidos de todos los usuarios con el usuario actual. Para esto pueden utilizarse diversas medidas como el Coeficiente de Correlación de Pearson, Correlación basadas en entropías o Correlación de Spearman. Seguidamente se lleva a cabo la selección de los vecinos donde se utilizan disímiles técnicas como Umbral de semejanza, Vecindad Centrada y Agregado de Vecinos. Finalmente se genera la recomendación que pudiera ser del interés del usuario (Nieto, 2007).

Por otra parte los algoritmos basados en elementos, en lugar de buscar similitudes entre usuarios buscan cercanía entre elementos. El procedimiento consiste en seleccionar los elementos que un usuario determinado ha votado y después comprobar cómo de similar es cada uno del resto de los elementos del sistema, para terminar recomendando los más parecidos. Existen distintas formas de evaluar la similitud entre elementos pero el procedimiento genérico consiste en tomar dos elementos x1, x2 y después calcular su similitud a partir de todos los usuarios que han votado ambos elementos. En teoría es la misma aproximación que la que se tenía con algoritmo basado en vecinos cercanos. A diferencia de estos últimos, se tiene en cuenta la similitud entre los elementos en lugar de los usuarios (Nieto, 2007).

Los Predictores Slope-One son una familia de algoritmos 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, mientras que su exactitud está a la par de algoritmos más complejos y costosos. A la hora del cálculo de la predicción para un usuario U, se tiene en cuenta tanto la información de usuarios que tienen en común la votación de algún elemento X como la información del resto de los elementos votados (Nieto, 2007).

1.1.3 Algoritmos para la búsqueda de la vecindad

La búsqueda de la vecindad es unos de los elementos más importantes en el tema de la recomendación por filtrado colaborativo porque según la cantidad de vecinos será la calidad de la recomendación. Existen diferentes técnicas para calcular la vecindad, por ejemplo (Torres, 2007):

- Utilización de un umbral.
- Elección de los mejores k vecinos (vecindad centrada).
- Agregado de Vecinos.

En la primera técnica se escoge un valor de umbral de manera empírica, eligiendo todos aquellos vecinos que lo superen. Esto puede presentar algunos problemas, como por ejemplo que no haya un número suficiente de vecinos que supere dicho umbral para valores de umbral elevados, provocando problemas de cobertura, o bien que el número de vecinos elegidos sea excesivamente grande, computacionalmente hablando, debido a un umbral bajo, haciendo inútil el intentar elegir un subconjunto de vecinos (Torres, 2007).

La segunda técnica consiste en elegir un número de K y utilizar los vecinos con mayor factor de similitud, evitando que haya pocos o muchos usuarios. La eleccion de este número K es crucial puesto que si se elige un valor demasiado grande habrá un número elevado de predicciones, perjudicando la exactitud de las mismas, en cambio, si el número es demasiado pequeño, la cobertura disminuirá y habrá muchas predicciones que no podrán realizarse (Torres, 2007).

Combinar ambas técnicas no mejora el uso del algoritmo KNN (por sus siglas en inglés *K-Nearest Neighbors*) por lo que la esencia está en encontrar un número adecuado de K en un entorno determinado (Torres, 2007).

Existe una tercera técnica para casos en los que la dispersión es acuciada[2]. Se trata de un agregado de vecinos en el que se utiliza el vecino más parecido al usuario activo, pasando a formar parte de la vecindad como integrante número uno. El resto de los K-1 vecinos se hallan en base al cálculo del centroide de la vecindad. Se entiende por centroide o pivote, el usuario autenticado al cual se le desea generar la recomendación (Torres, 2007). Seguidamente se calcula la similitud que existe entre el pivote y los restantes usuarios de la base de datos. Posteriormente se calcula un promedio entre todas las similitudes encontradas. Esto le permite a los usuarios que estén por encima del promedio calculado, tomar parte en la formación de dicha vecindad, lo que puede ser beneficioso para conjuntos de datos dispersos (Sarwar, 2000).

Después de un profundo estudio se concluye que la técnica que se va a utilizar es el agregado de vecinos ya que es la que más se adapta, teniendo en cuenta la variabilidad de los ambientes donde se va a desplegar la plataforma AGORAV.

[2] Se refiere a los casos en los que la dispersión entre el usuario centroide de la vecindad y el resto de los vecinos es demasiada grande.

1.2 Descripción del Sistema de Recomendación AGORAV

Las recomendaciones de materiales multimedia de preferencia para el usuario surgen a partir del aumento del consumo audiovisual y la dispersión de este tipo de materiales en la red. Los sistemas de recomendación son utilizados como herramientas para la automatización de este proceso. Como valor agregado, la plataforma AGORAV cuenta con *plugins* que permiten la recomendación de contenido audiovisual, los cuales utilizan las técnicas filtrado colaborativo y basado en contenido para su implementación. En la actualidad, la plataforma AGORAV cuenta con un *plugin* de recomendación, el cual realiza la selección de sugerencias, a partir de un tamaño fijo de vecinos, lo cual trae consigo una serie de inconvenientes cuando se necesita aplicar en entornos sumamente cambiantes, debido a la variabilidad que puede existir en lo que respecta a la cantidad de usuarios.

Lo anteriormente planteado corrobora la necesidad de desarrollar un *plugin*, el cual utilice una cantidad variable de colaboradores, debido a que en dependencia del entorno en que sea aplicable dicho *plugin*, deberá ser la cantidad de usuarios con los cuales comparar a un usuario determinado. De esta manera se garantiza que la muestra de usuarios a seleccionar sea representativa en correspondencia con el universo al que estos pertenezcan.

1.3 Metodología de Desarrollo

Una buena selección tanto de las herramientas a utilizar como de la metodología que se desea emplear ayuda en cuanto a la calidad del software deseado, pues de ahí se deriva el papel que debe desempeñar cada miembro dentro del equipo de desarrollo y qué actividades tienen que cumplir, se definen qué artefactos deben ser creados y se pormenoriza cada detalle de la información del producto que se alcance, como resultado de toda la actividad realizada (Jiménez, 2013).

1.3.1 Metodología AUP

Como metodología de desarrollo de software se selecciona AUP (por sus siglas en inglés *Agile Unified Process*) que es una versión simplificada de RUP (por sus siglas en inglés *Rational Unified Process*) (Palacio, 2008). Se selecciona esta metodología ya que es actualmente la que se está utilizando en la plataforma AGORAV, para entonces tener en cuenta la compatibilidad entre los artefactos generados y los existentes en la plataforma. Además también brinda la posibilidad de generar solamente la documentación más importante para el proyecto y la necesaria para que pueda ser entendida por otros desarrolladores para

trabajar en versiones avanzadas del *plugin*. Dentro de sus características principales también destaca que presenta un desarrollo iterativo incremental, lo que permite que a medida que se generen artefactos, se puedan integrar a la documentación del proyecto.

1.4 Lenguaje de Modelo Unificado

Lenguaje de Modelado Unificado (por sus siglas en inglés UML: *Unified Modeling Language*) es un lenguaje para visualizar, especificar, construir y documentar los artefactos que se crean durante el proceso de desarrollo. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos (Jacobson, 2000). En el desarrollo de este proyecto para poder modelar y diseñar los principales artefactos generados en el proceso de desarrollo del *plugin,* y para lograr un lenguaje común que pueda ser entendido por cualquier desarrollador, se usará como lenguaje de representación visual UML.

1.5 Herramientas CASE

Se puede definir a las herramientas CASE (por sus siglas en inglés *Computer Aided Software Engineering*) como un conjunto de programas y ayudas que dan asistencia a los desarrolladores de software durante todo el ciclo de vida del mismo. Las herramientas CASE facilitan el proceso de revisión ya que proporcionan bases para las definiciones y estándares para los datos (Sierra, 2008).

1.5.1 Visual Paradigm

Se consideró factible el uso del *Visual Paradigm* como herramienta CASE porque presenta una serie de ventajas entre las que se encuentran que soporta el ciclo de vida completo del desarrollo del software, análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Además se necesitan modelar los artefactos para cada una de las faces por las que transcurra el desarrollo del *plugin*, para asegurar el entendimiento para el proyecto que hará uso de la solución. También posibilita la exportación de los diagramas en imágenes jpg, png y svg .Además entre sus características están que es multiplataforma y emplea las últimas notaciones de UML.

1.6 Lenguaje de Programación

Luego de tener un diseño de la solución propuesta se puede pasar a la implementación del sistema, para esto se hace necesario elegir el lenguaje de programación a utilizar. Un lenguaje de programación es una herramienta que permite comunicar e instruir a la computadora para que realice una tarea específica (Lobo, 2005).

Se seleccionó el lenguaje de programación C++ para el desarrollo de esta investigación debido a que presenta una serie de ventajas sumamente destacables entre las que se encuentran que es un lenguaje compilado y no interpretado, lo cual lo convierte en un lenguaje rápido, y el trabajo que realiza con punteros lo que permite una mejor gestión de la memoria. Además el Sistema Gestor de Procesos de Medias (SGPM) y los *plugins* que con él se relacionan están contenidos en dicho lenguaje y para lograr una adecuada compatibilidad se recomienda que el *plugin* también sea programado en C++.

1.7 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (por sus siglas en inglés IDE: Integrated Development Environment) es un programa compuesto por un conjunto de herramientas que son de utilidad para el programador. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (Alegsa, 2010).

1.7.1 QtCreator 3.1.1

QtCreator 3.1.1 es un IDE multi-plataforma, que permite a los desarrolladores crear aplicaciones para escritorio. Entre las principales características de QtCreator 3.1.1 se puede encontrar que posee un avanzado editor de código C++, depurador visual, resaltado y autocompletado de código. Tiene la posibilidad de generar todos los archivos necesarios si se importa un proyecto existente o si se desea crear uno desde cero (Wiki, 2015). Basado en las características y las ventajas vistas anteriormente y teniendo en cuenta que el lenguaje de programación utilizado para el *plugin* es C++ se seleccionó este IDE para el desarrollo del mismo.

1.7.2 Framework QT

Qt es una framework multiplataforma en C++ de desarrollo de aplicaciones. Se utiliza fundamentalmente para desarrollar aplicaciones con interfaz gráfica, gracias al conjunto de controles independientes de la plataforma que ofrece, aunque también es usado para crear herramientas de línea de comando o consolas de gestión para servicios. (Torres, 2014).

1.8 Sistema Gestor de Base de Datos

Un Sistema Gestor de Base de Datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos (Álvarez, 2012). Ayuda a realizar las siguientes acciones:

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.
- Control de la seguridad y privacidad de los datos y la manipulación de los datos.

1.8.1 PostgresSQL

Para el buen funcionamiento de la aplicación se necesita contar con un sistema gestor de base de datos que posibilite la obtención de las preferencias de los usuarios de la forma más sencilla y rápida posible. En la actualidad PostgresSQL es considerado uno de los gestores de bases de datos de código abierto, debido a que soporta casi toda la sintaxis SQL (incluyendo subconsultas, transacciones y funciones definidas por el usuario) (Lockhart, 2009). Tomando como referencia las características antes expuestas y por la condición de ser una herramienta libre y multiplataforma, se decide utilizar PostgresSQL como Sistema Gestor de Base de Datos (SGBD) en el desarrollo del sistema.

1.9 Conclusiones Parciales

Al término de este capítulo se concluye que la plataforma AGORAV necesita de un componente que sea capaz de arrojar una variabilidad en la cantidad de usuarios a recomendar. El *plugin* existente utiliza una cantidad fija de colaboradores, y dado a que, las variables que afectan directamente al sistema de recomendación son muy cambiantes en los distintos negocios a los que está destinada la plataforma AGORAV, se hace necesario el desarrollo de un componente que utilice una cantidad variable de usuarios. En dependencia del entorno en que sea aplicado dicho *plugin*, deberá ser la cantidad de usuarios a recomendar. Para el desarrollo de la solución al problema planteado, se utilizará la técnica agregado de vecinos como algoritmo para la búsqueda de la vecindad.

En este capítulo además se determinó, según las necesidades de la solución, utilizar como metodología de desarrollo AUP; C++ como lenguaje de programación, como IDE se utilizara QtCreator V3.1.1, como framework de desarrollo se utilizara Qt V5.2.1, PostgresSQL como SGBD, UML como lenguaje de modelado y *Visual Paradigm* como herramienta CASE.

Capítulo 2: Análisis y diseño del componente para el cálculo de los k vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo de la plataforma AGORAV

Este capítulo tiene como objetivos presentar la descripción del modelo de dominio, especificar los requisitos funcionales y no funcionales del componente a desarrollar. Se define el diagrama de caso de uso del sistema con la descripción de cada caso de uso, además de especificar los estilos, patrones arquitectónicos y de diseño que se utilizaron en la solución al problema planteado anteriormente.

2.1 Modelo de Dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. En él se muestran las clases conceptuales o vocabulario del dominio y como se relacionan unas con otras mediante asociaciones. Este modelo facilita la comunicación entre desarrolladores y empresas brindando un lenguaje común, que a su vez, hace mucho más sencilla la comunicación de los requisitos y el mantenimiento de la aplicación. El objetivo principal del modelo es comprender y describir las clases más importantes dentro del contexto del sistema (Larman, 1999). Para la realización del sistema se decide hacer uso de un modelo de dominio, producto a que no se lograron identificar procesos claros y bien estructurados que permitan realizar el modelado completo del negocio.

2.1.1 Descripción de clases del modelo de dominio

El usuario autenticado accede a la plataforma con el fin de visualizar unos de los archivos multimedia con los que cuenta la plataforma. La plataforma realiza una petición de recomendación al Gestor de Proceso y el Sistema de Recomendación mediante el SGPM genera una recomendación para el usuario autenticado.

Usuario: Usuario que accede a la plataforma con el fin de visualizar algún contenido multimedia.

Archivo Multimedia: No son más que los archivos que son publicados en la plataforma tales como: películas, series, documentales.

SGPM: El Sistema Gestor de Procesos de Media está compuesto por dos subsistemas, el Gestor de Procesos de Media y la Plataforma de Codificación e Indexación. El SGPM publica un servicio para brindar las funcionalidades existentes, las que pueden ser solicitadas por varias aplicaciones. Esta tecnología atiende las peticiones y a medida que se van ejecutando las tareas notifica el estado a los responsables de

las mismas. En la Plataforma de Codificación e Indexación se realizan tareas de procesamiento de medias. Los procesos que se realizan sobre las medias son codificación, indexación, transferencia y actualización de metadatos. La Plataforma de Codificación e Indexación permite que se le adicionen varias operaciones mediante el uso de *plugins*, facilitando grandes niveles de escalabilidad y una elevada posibilidad de integración (Viñolo, y otros, 2012).

Sistema de Recomendación: *Plugin* encargado de hacer las recomendaciones por filtrado colaborativo.

Plataforma: Plataforma de publicación de contenido multimedia AGORAV.

2.1.2 Diagrama de Modelo de Dominio

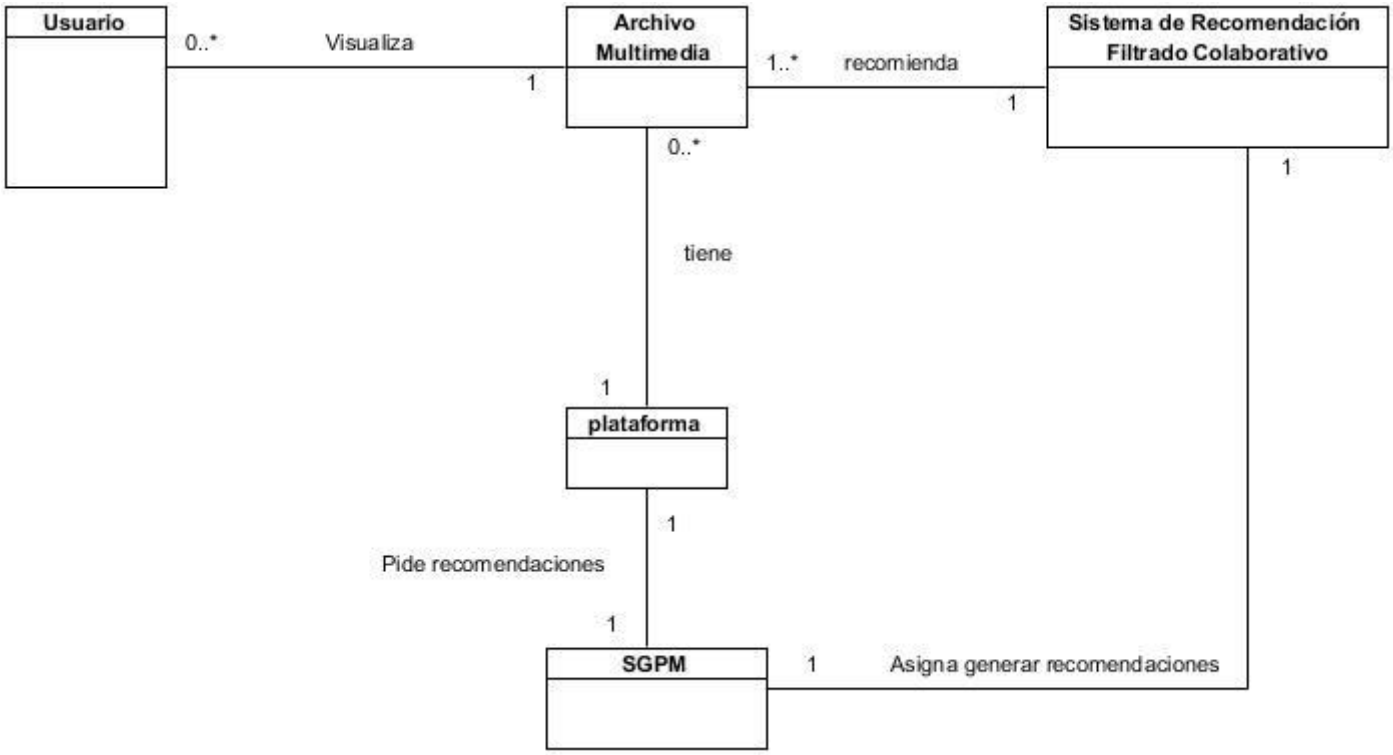

Fig. 1: Diagrama de modelo de dominio.

2.2 Requisitos de software

La ingeniería de requisitos cumple un papel primordial en el proceso de producción de software, debido a que va enfocada a un área fundamental en el desarrollo de todo sistema, que es la definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta el comportamiento del sistema, de manera que le garantice minimizar los problemas relacionados al desarrollo de sistemas. Los requisitos se encuentran divididos en dos grandes grupos: los funcionales y los no funcionales (Cedeño, 2011).

2.2.1 Requisitos Funcionales

Describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema (Quiroga, 2010).

RF1: Garantizar que el *plugin* sea integrable con el SGPM

Descripción: El sistema debe ser capaz de integrarse con el SGPM.

Entradas: No procede.

Salida: Interfaz intermedia que garantiza el acceso a las funcionalidades del *plugin*.

RF2: Obtener la cantidad de usuarios existentes

Descripción: El sistema debe permitir conocer cuántos usuarios existen en la base de datos ya que esto es necesario para calcular las similitudes en el método Calcular_Similitud_Pearson.

Entrada: Se necesita conocer el promedio de votaciones de todos los usuarios así como la cantidad de publicaciones que estos han visto.

Salida: La cantidad de usuarios existentes en la Base de Datos.

RF3: Obtener la cantidad de medias existentes

Descripción: El sistema debe permitir conocer cuántas medias existen en la Base de Datos ya que esto es primordial a la hora de calcular la similitud del usuario activo con el resto de los usuarios de la Base de Datos.

Entrada: Se necesita conocer la cantidad de medias que cada usuario visualizó.

Salida: La cantidad de medias existentes en la Base de Datos.

RF4: Obtener las preferencias

Descripción: El sistema debe permitir conocer las preferencias de los usuarios existentes en la Base de Datos.

Entrada: Se necesita calcular la similitud basada en la preferencia de los usuarios.

Salida: Preferencia de los usuarios.

RF5: Buscar usuario de mayor similitud al usuario activo

Descripción: El sistema debe ser capaz de calcular cuál es el usuario que tiene mayor parentesco al usuario activo.

Entrada: Lista de similitudes.

Salida: Usuario con mayor similitud al usuario activo.

RF6: Buscar el promedio de similitudes

Descripción: El sistema debe ser capaz de calcular el promedio de las similitudes, para en base a este ir formando la vecindad.

Entrada: Lista de similitud.

Salida: Valor promedio.

RF7: Calcular la similitud entre el usuario activo y los restantes usuarios de la BD

Descripción: El sistema debe ser capaz de buscar la similitud entre el usuario activo y los demás usuarios existen en la base de datos para así conformar la vecindad con aquellos que superen el promedio de similitudes calculado con anterioridad.

Entrada: Usuario activo.

Salida: Grupo de usuarios.

2.2.2 Requisitos No Funcionales

Describen aspectos del sistema que son visibles por el usuario y que no incluyen una relación directa con el comportamiento funcional del sistema. Los requerimientos no funcionales incluyen restricciones como son: Diseño, Implementación y Software (Quiroga, 2010).

RNF1: Requisitos de Hardware

Descripción: Para garantizar el correcto funcionamiento del componente se debe contar con los siguientes elementos:

- Procesador Intel Core i3 1.60GHz o superior.
- Memoria RAM DDR de 2 Gb o superior.
- Disco duro de 160 Gb o superior.

RNF2: Restricciones de Diseño

Descripción: Se deben respetar los estándares definidos por el *framework* QT para la codificación del código, además de utilizar el estándar de codificación UTF-8 para la creación de tablas en la base de datos en PostgresSQL.

RNF3: Restricciones de Implementación

Descripción: Utilizar C++ con el objetivo de mantener la compatibilidad con el SGPM y el *plugin* de recomendación.

RNF4: Requisitos de Software

Descripción: Se requiere un SGBD PostgresSQL 9.4 y también un servidor con el SGPM instalado.

2.3 Diagrama de Caso de Uso del Sistema

El diagrama de Caso de Uso del Sistema muestra la relación que existe entre los actores y los casos de uso del sistema, donde cada caso de uso representa una funcionalidad. Sirve como contrato entre los clientes y los desarrolladores. Se utiliza como punto de partida inicial para llegar a las actividades de análisis y diseño que posteriormente se realizarán en el desarrollo del software (Ivar Jacobson, 2000).

Fig. 2: Diagrama de caso de uso del sistema.

Tabla 1: Descripción textual CU Generar tamaño de la vecindad.

Objetivo	Generar tamaño de la vecindad.	
Actor	Sistema Gestor de Proceso de Medias.	
Resumen	El CU inicia cuando el sistema necesita generar una recomendación sobre cualquiera usuario y para esto necesita una vecindad con la cual comparar el mismo. El CU termina cuando se ha generado una vecindad y se le puede hacer una recomendación al usuario.	
Complejidad	Alta.	
Prioridad	Crítico.	
Precondiciones	El usuario está autenticado y tiene historial en la base de datos. No existen recomendaciones válidas en la base de dato para este usuario.	
Postcondiciones	Se guarda el tamaño de la vecindad.	
Flujo de eventos		
Flujo básico: Generar tamaño de la vecindad		
	Actor	**Sistema**
1.	Solicita al *plugin* desarrollado.	
2.		Obtiene los datos relacionados al historial de usuario existente en la base de datos.
3.		Calcula similitud entre el usuario activo y el resto de los usuarios de la base de datos.

4.		Se conforma la vecindad, para esto se calcula el promedio de las similitudes de todos los usuarios y los usuarios que estén por encima de este promedio pasan a formar parte de la vecindad.
5.		Se envía el tamaño de la vecindad al gestor de proceso de medias a través del protocolo XMLRPC 777. Termina el caso de uso.

Relaciones	**CU Incluidos**	No tiene.
	CU Extendidos	No tiene.
Requisitos no funcionales	RNF2 Restricciones de diseño, RNF3 Restricciones de implementación.	
Asuntos pendientes	No existe.	

2.4 Estilos Arquitectónicos

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón, las restricciones en la forma y la estructura de un grupo de numerosos y variados sistemas de software (Ciencias, 2008-2009). Según Pressman *"Cada estilo describe una categoría del sistema que contiene: un conjunto de componentes por ejemplo, una base de datos, módulos computacionales que realizan una función requerida por el sistema; un conjunto de conectores que posibilitan la comunicación, la coordinación y la cooperación entre los componentes; restricciones que definen como se pueden integrar los componentes que forman el sistema; y modelos semánticos que permiten al diseñador entender las*

propiedades globales de un sistema para analizar las propiedades conocidas de sus partes constituyentes" (Pressman, 2010).

Importancia de los estilos:

- Sirven para sintetizar y tener un lenguaje que describa la estructura de las soluciones.
- Pocos estilos abstractos encapsulan una enorme variedad de configuraciones concretas.
- Definen los patrones posibles de las aplicaciones.

El estilo arquitectónico seleccionado en esta investigación es el de llamada y retorno debido a que los principales elementos de la arquitectura estarán centrados hacia los objetos y serán estos las unidades de modelado a partir de las clases que los definen interactuando a través de funciones y métodos. Estos estilos son los más generalizados en sistemas a gran escala y enfatizan en la escalabilidad (Carlos Kicillof, 2004). El estilo seleccionado se evidencia en el flujo que existe entre el SGPM y el *plugin* a desarrollar, cuando el gestor hace la llamada al *plugin* y este se ejecuta retornando el tamaño de la vecindad.

2.4.1 Patrones arquitectónicos

Un patrón arquitectónico es una solución a un problema en un contexto, codifica conocimiento específico acumulado por la experiencia en un dominio. Un sistema bien estructurado está lleno de patrones. Cada patrón describe un problema que ocurre una y otra vez en el ambiente, y luego describe el núcleo de la solución a ese problema, de tal manera que puede utilizarse esa solución varias veces más, sin hacer la misma cosa dos veces (Ciencias, 2008-2009).

Existen diferentes categorías de patrones según la escala o nivel de abstracción:

- Patrones arquitecturales: Aquellos que expresan un esquema organizativo estructural fundamental para sistema de software.
- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño.
- Idiomas: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

Para el desarrollo de este *plugin* se utilizara el patrón arquitectónico N-Capas, específicamente posee tres capaz: Acceso a datos, Lógica de Negocio y Presentación. Cada una de estas capas posee un conjunto de funcionalidades relacionadas entre ellas y que en conjunto reflejan el *plugin* en su totalidad. A continuación se describe cada una de estas partes del producto.

La capa de **Acceso a Datos** permite acceder a todos los datos necesarios que se encuentran en la base de datos historial, también cuenta con métodos que permiten la conexión, consultas a la base de datos y desconexión. Por otra parte la capa de **Lógica de Negocio** cuenta con la clase CL_Vecindad que es donde se encuentran los métodos principales del *plugin* como por ejemplo: Buscar_Mayor, Promedio, Calcular_Similitud_Pearson. La capa de **Presentación** contiene la interfaz a través de la cual el SGPM podrá interactuar con el *plugin*.

2.4.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboradores y la distribución de responsabilidades (Ciencias, 2008-2009).

Los patrones de diseño se encuentran divididos en dos grupos, los GRASP (*General Responsibility Assignment Software Patterns o Patrones Generales de Asignación de Responsabilidad de Software*) que establecen los principios fundamentales de la asignación de responsabilidades a objetos y los llamados GOF (*Gang of Four o Banda de los Cuatro*).

Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (Larman, 1999).

Experto: El experto en información se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos (Larman, 1999). Asigna una responsabilidad a la clase que cuenta con la información necesaria para cumplirla. Constituye un principio básico que frecuentemente se utiliza en el diseño orientado a objetos Este patrón se evidencia en la CL_Vecindad que es donde se lleva a cabo todo el procesamiento.

Controlador: Permite asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase. Esté se evidencia en la clase CL_Vecindad del *plugin*.

Bajo acoplamiento: Su principal objetivo es lograr poca dependencia entre clases, con esto se reduce el impacto de los cambios en otros componentes. Permite asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible. El uso de este patrón se encuentra

reflejado en todas las clases del *plugin*, de manera tal que cuando se produzca alguna modificación en algunas de ellas la repercusión en las otras será mínima.

Alta cohesión: Asignar una responsabilidad de modo que la cohesión siga siendo alta. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme o excesivo (Larman, 1999). La utilización de este patrón se encuentra presente en la definición de tareas y responsabilidades de cada clase en el *plugin*.

Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del patrón creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación (Larman, 1999). Un ejemplo de donde se pone de manifiesto este patrón es en la clase CL_Vecindad a la hora de crear los objetos con la información pertinente en cada caso.

Patrones GOF

Los patrones GOF son soluciones técnicas basadas en Programación Orientada a Objetos (POO). Están clasificados en patrones creacionales, estructurales y de comportamiento. Los creacionales son los encargados de la creación de instancias de los objetos, los estructurales plantean las relaciones entre clases y los de comportamiento plantean la interacción y cooperación entre las clases (Larman, 1999).

Iterator: Este patrón se basa en el recorrido de todos los elementos de una colección, una vez cada uno, y sin seguir un orden en concreto. (Mondaray, 2012). Este patrón se utilizó en el método Buscar_Mayor a la hora de recorrer el arreglo para buscar el usuario de mayor similitud al usuario actual, y también en el método Promedio para calcular el promedio del arreglo de similitudes de los usuarios.

Strategy: Es el primer patrón de los que vemos que se centra en el polimorfismo. Determina como se debe realizar el intercambio de mensajes entre diferentes objetos para resolver una tarea. Mantiene un conjunto de algoritmos de entre los cuales el objeto cliente puede elegir aquel que le conviene e intercambiarlo dinámicamente según sus necesidades (Mondaray, 2012). Este patrón se utilizó para enviar la cantidad de vecinos al SGPM.

Singleton: Es un patrón de creación y garantiza que una clase posea una única instancia, a la vez que provee un punto de acceso global a ella. Permite por su diseño restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Se evidencia su uso en el objeto de acceso a datos del *plugin*.

***Interface*:** Este patrón permite la creación de una interfaz para facilitar el acceso a las funcionalidades del *plugin*.

2.5 Diagrama de clases de Diseño

El Diagrama de Clase es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema se especifica, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama, y se modifica para satisfacer los detalles de las implementaciones.

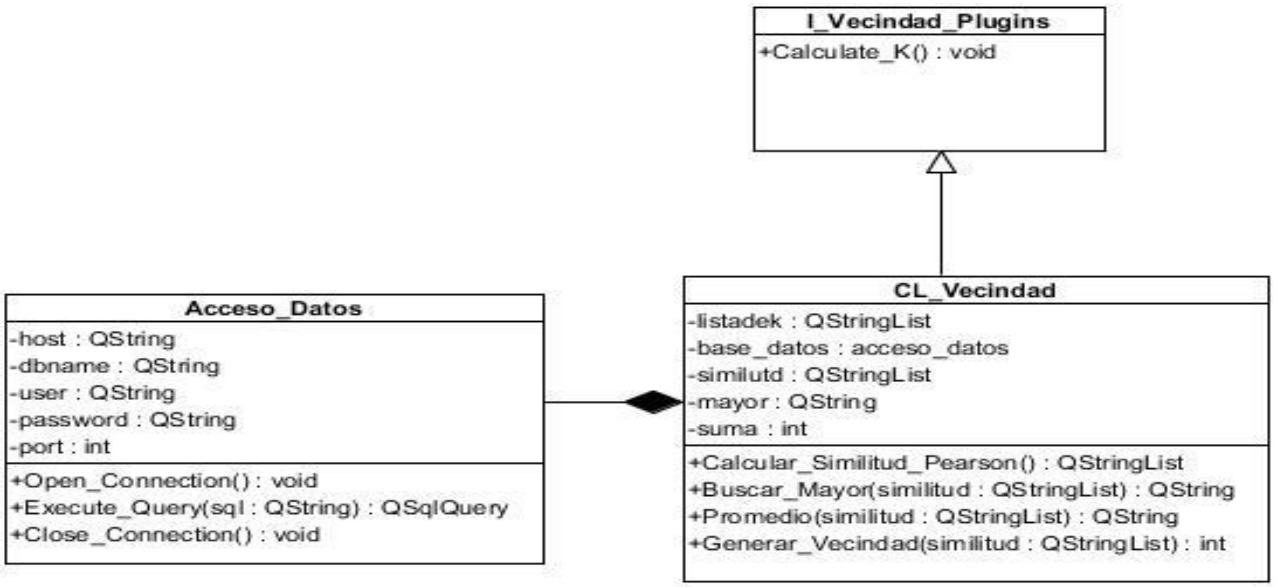

Fig. 3: Diagrama de clase de diseño.

2.5.1 Descripción de las clases del diseño

CL_Vecindad: es la clase principal del *plugin*, en ella se llevan a cabo todos los métodos principales del *plugin*, a través de ella también se puede acceder a todos los datos necesarios que se encuentran en la base de datos. Esta clase también es la encarga de generar el número final de vecinos.

Acceso_Datos: contiene todas las funciones de acceso a los datos almacenados físicamente en la base de datos, además permite la conexión con la misma.

I_Vecindad_Plugins: es la interfaz que permite la conexión del *plugin* con el GPM para a través de este comunicarse con el Sistema de Recomendación.

2.6 Diagrama de Secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo. Esta descripción es importante porque puede dar detalles a los casos de uso, aclarándolos al nivel de mensajes de los objetos existentes, como también muestra el uso de los mensajes de las clases diseñadas en el contexto de una operación.

En la imagen se muestra el diagrama de secuencia del caso de uso Generar tamaño de la Vecindad:

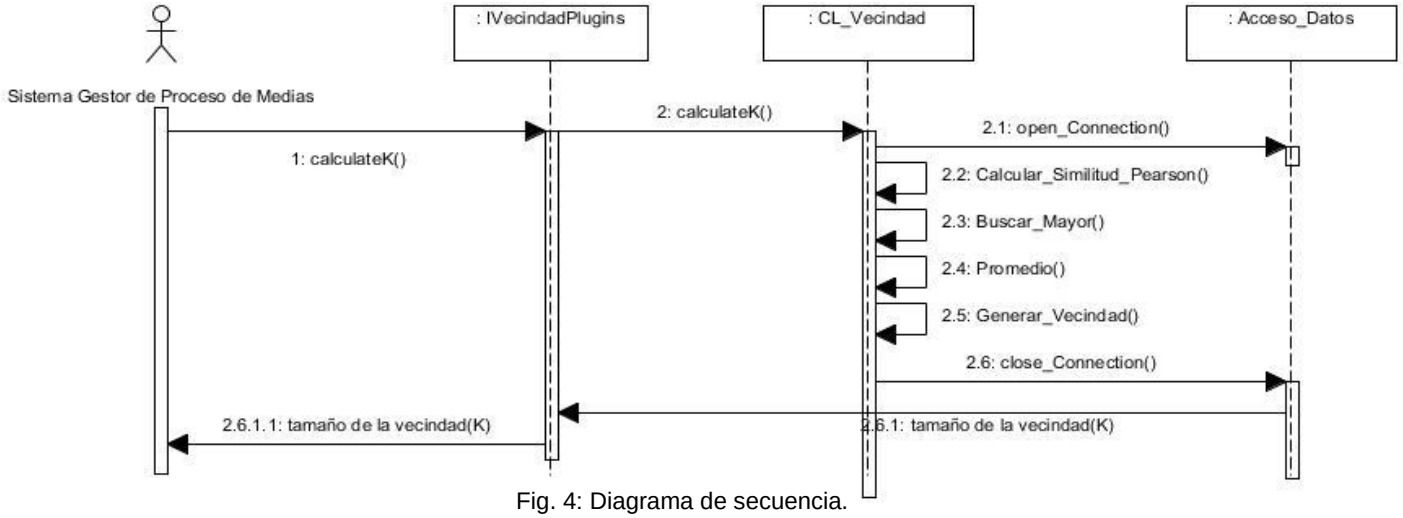

Fig. 4: Diagrama de secuencia.

2.7 Conclusiones Parciales

La realización del modelo de diseño y los diagramas asociados a este sirvieron de abstracción entre la implementación y el sistema, razón por lo que fue utilizado como entrada fundamental de las actividades de implementación. Por otra parte se logró identificar una serie de requisitos funcionales y no funcionales de vital importancia para el correcto funcionamiento del sistema. Junto con esto y una correcta implementación de los mismos, se garantiza minimizar los problemas relacionados al desarrollo del *plugin* y a su funcionamiento. También se identificaron los casos de uso por los que debe estar integrado el sistema a desarrollar y se realizó una descripción para cada uno de ellos. Además se concluyó que lo más factible en la construcción del *plugin* es hacer uso del patrón arquitectónico N-Capas, debido a las facilidades que brinda la desagregación de las capas definidas del *plugin*. Los estilos y patrones utilizados proporcionaron uniformidad, comprensión y escalabilidad a la solución desarrollada, al mismo tiempo que favorecen la rigurosidad en su diseño.

Capítulo 3: Implementación y pruebas del componente para el cálculo de los k vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo

En el presente capítulo se abordan aspectos referentes a la implementación y las pruebas que se realizarán al sistema en desarrollo. Se hace referencia al modelo de datos, que permitirá adquirir una mayor visión de las tablas de la base de datos necesarias para la interacción con el sistema. También se construye el modelo de implementación y se modela el diagrama de componentes que muestra las organizaciones y las dependencias lógicas entre los elementos del sistema. Además se realiza el modelo de despliegue que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre sus nodos. Finalmente para verificar la calidad y el adecuado funcionamiento del software, realizando las pruebas pertinentes utilizando, técnicas que guíen el proceso de prueba.

3.1 Modelo de Datos

Para el cálculo de los k vecinos más cercanos no fue necesario crear tablas, por eso a continuación son mostradas en el modelo de datos las tablas ya existentes en la plataforma AGORAV, y que se utilizan para obtener los datos necesarios para que el algoritmo arroje los resultados esperados.

Listado de tablas:

- node
- datos_publicaciones
- historial
- archivo_multimedia
- sistema_recomendacion_fc

Fig. 5: Modelo de datos *(Limonta, 2014)*.

A continuación se describen las tablas utilizadas de la plataforma AGORAV:

Tabla 2: Descripción de la tabla historial.

Nombre: historial.
Descripción: En esta tabla se almacena la información de los archivos multimedia que cada usuario ha visualizado en la plataforma.

Atributo	Tipo	Descripción
id_historial	Int4	Valor numérico incremental que identifica todos los campos de la tabla.
id_usuario	Int4	Valor numérico que identifica al usuario.
id_publicacion	Int4	Valor numérico que identifica al archivo multimedia publicado que el usuario visualizó.
votación	Int4	Votación que el usuario le dio al archivo multimedia visualizado.
usuario	Int4	Valor numérico que identifica al usuario
fecha	Int4	Fecha en la que el usuario visualizó el archivo multimedia convertida a un número entero.

Tabla 3: Descripción de la tabla sistema_recomendacion_fc.

Nombre: sistema_recomendacion_fc.		
Descripción: En esta tabla se almacenan las recomendaciones realizadas a los usuarios de la plataforma.		
Atributo	**Tipo**	**Descripción**
id	Int4	Valor numérico incremental que identifica todos los campos de la tabla.
am_id	Int4	Valor numérico que identifica al archivo multimedia recomendado al usuario.
fecha	Int4	Fecha en la que se le realizó la recomendación al usuario, convertida a un número entero.
id_usuario	Int4	Valor numérico que identifica al usuario que se le realizó la recomendación.

3.2 Modelo de Implementación

El modelo de implementación describe como los elementos del modelo de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente, ejecutables, etc. El modelo de

implementación describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y como dependen los componentes unos de otros (Jacobson, 2000).

3.2.1 Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos y sus relaciones en el entorno. Los componentes representan los tipos de elementos de software que entran en la elaboración de aplicaciones informáticas, y las relaciones de dependencia se utilizan para indicar que un componente utiliza los servicios ofrecidos por otro componente.

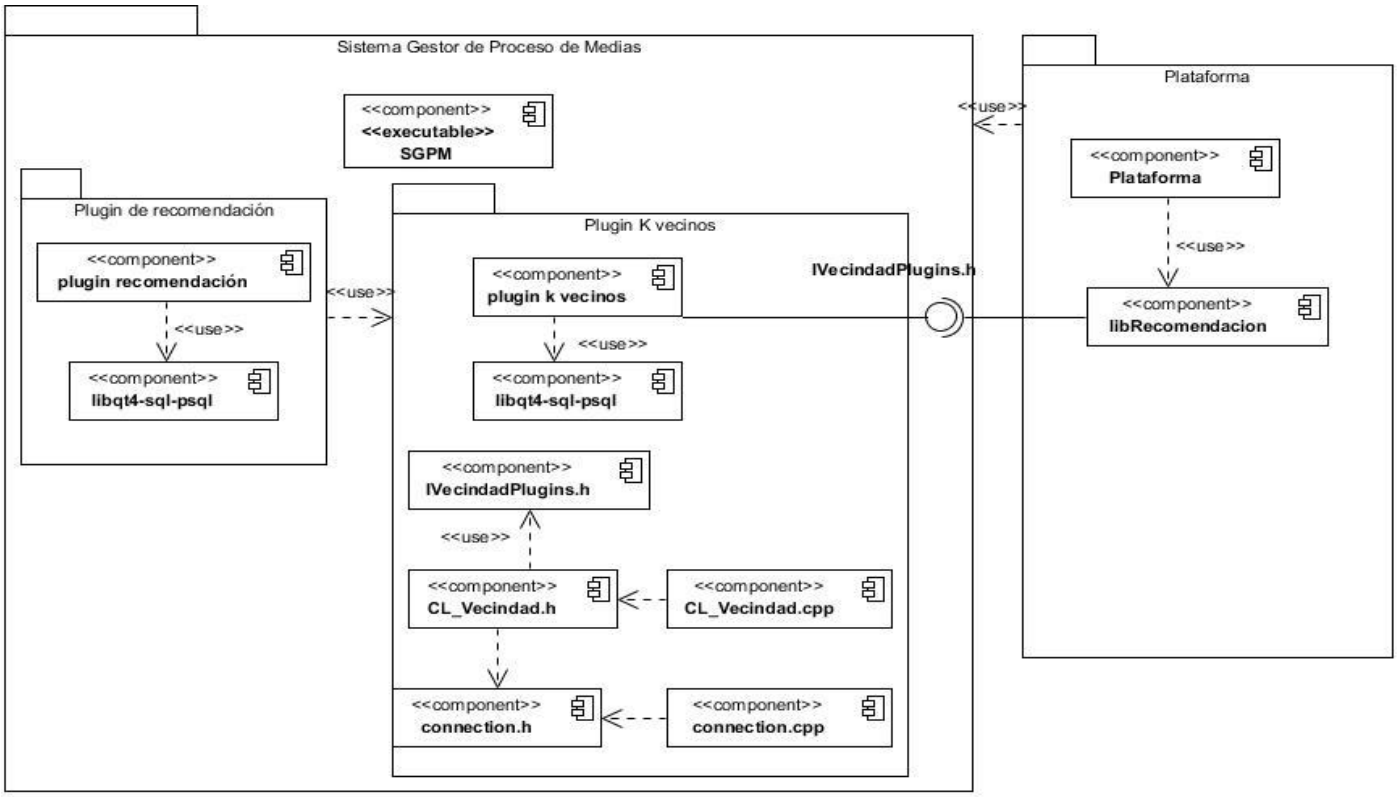

Fig. 6: Diagrama de componentes.

El diagrama de componentes que se muestra contiene los siguientes ficheros:

- **plugin K vecinos**: encargado de generar la vecindad que se va usar para generar la recomendación.
- **plugin recomendación:** encargado de generar la recomendación.
- **libqt4-sql-psql**: librería utilizada por el Framework Qt para la comunicación con la base de datos.

3.2.2 Modelo de despliegue

El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo hardware similar. Puede describir diferentes configuraciones de red, incluidas las configuraciones para prueba y para simulación (Ivar Jacobson, 2000).

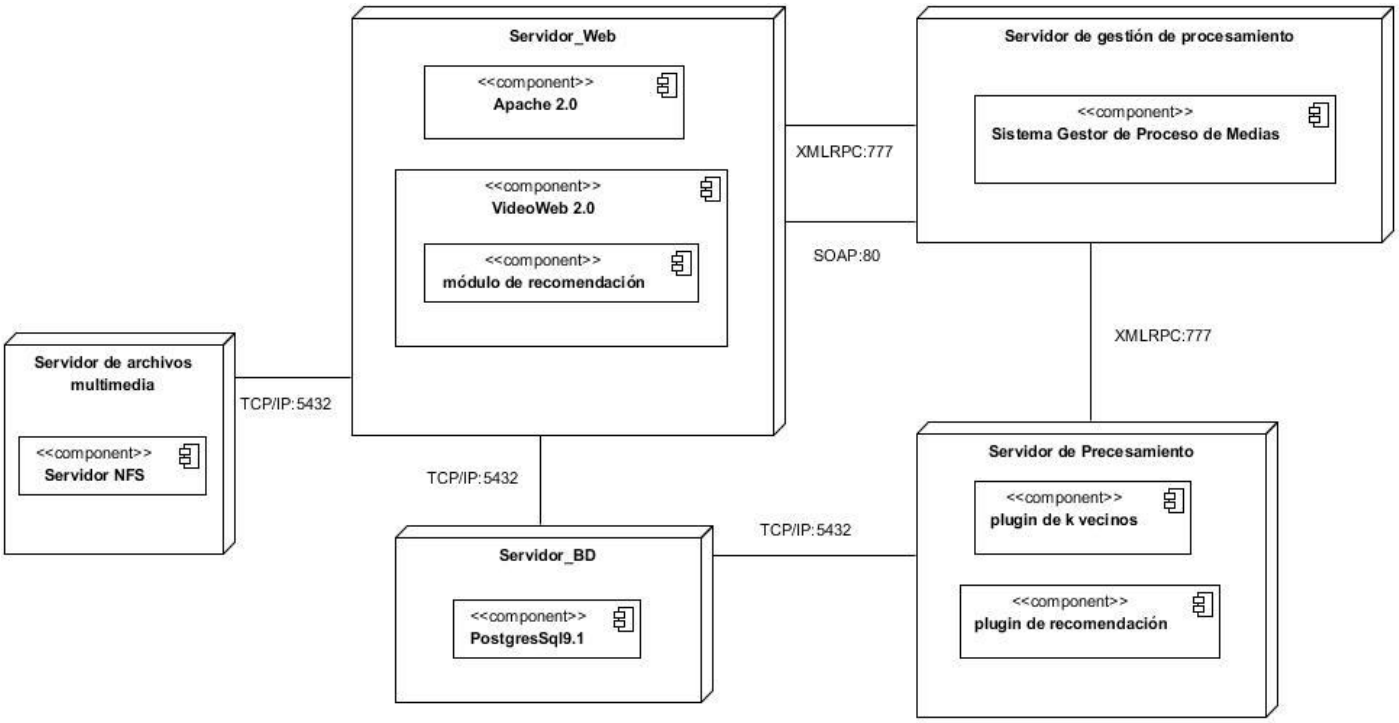

Fig. 7: Diagrama de despliegue.

Descripción de los nodos:

Servidor Web: Nodo servidor con el cual se comunican las PC mediante el protocolo HTTP (Hypertext Transfer Protocol). Aquí se encuentra instalado el servidor web Apache.

Servidor_BD: Nodo servidor de base de datos al cual se conecta el servidor web a través del protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) por el puerto 5432.

Sistema Gestor de Procesos de Medias: Nodo servidor que se comunica con el servidor web para la generación de recomendaciones, esta comunicación se realiza a través del protocolo SOAP (Simple Object

Access Protocol) por el puerto 8080, mientras que viceversa ser realiza a través del protocolo XMLRPC por el puerto 777.

Servidor de procesamiento: Nodo servidor que se comunica con el Sistema Gestor de Procesos de Medias para generar las recomendaciones .Estos servidores de procesamiento necesitan para la generación de recomendaciones los datos almacenados en la base de datos, es por esto que se comunican con el servidor de base de datos y lo hace a través del protocolo TCP/IP por el puerto 5432.

Servidor de Archivos Multimedia: Nodo servidor que se comunica con el servidor web a través del protocolo TCP/IP (Transmission Control Protocol/Internet Protocol) por el puerto 5432.

3.3 Estándar de Codificación

Un estándar completo de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, este debe permitir definir una estandarización válida para las clases y los atributos, que otros programadores comprendan la codificación, que el uso de comentarios les sirva para un mejor entendimiento del código. Un código fuente completo debe reflejar un estilo homogéneo, como si un único programador hubiese escrito todo el código de una sola vez. La legibilidad del código fuente repercute directamente en lo bien que un programador comprende un sistema de software. El mantenimiento del código es la facilidad con que el sistema de software puede modificarse para añadirle nuevas características, modificar las ya existentes, depurar errores, o mejorar el rendimiento (2011). En el desarrollo del *plugin* se utilizo el estándar de codificación C++ v1.1 el cual cuenta con una serie de principios generales como son:

- Los nombres de cada uno de los elementos del programa deben ser significativos; su nombre debe explicar en lo posible el uso del elemento.
- No manejar en los programas más de una instrucción por línea.
- Declarar las variables en líneas separadas.
- Añadir comentarios descriptivos junto a cada declaración de variables, si es necesario.
- La mayoría de los elementos se deben nombrar usando sustantivos.
- Los atributos deben comenzar con letra minúsculas y los métodos deben comenzar con letra mayúsculas.

3.3.1 Estilos de Codificación

Los estilos de codificación se utilizan de manera uniforme en todo el proceso de implementación del sistema para facilitarle la lectura y comprensión del código a aquella persona que necesite realizar alguna modificación, por eso es importante respetar estándares de codificación oficiales o no, pero que se siga un estándar y lograr que la estructura del código fuente refleje la estructura lógica del programa.

Para garantizar la uniformidad y la legibilidad del código, y con estos la calidad de la solución que se implementará, se decidió utilizar el estilo Allman. Este estilo plantea que se debe de crear una nueva línea para las llaves, e identar el código debajo de ellas. La llave de cierre del bloque debe estar identada de la misma forma que la de inicio es una de las codificaciones, siendo más fácil encontrar donde comienza y termina cada bloque. En la siguiente figura se hace una representación del código siguiendo el estilo de codificación seleccionado.

```
QString CL_Vecindad::Buscar_mayor(QStringList similitud)
{
    int mayor_similitud=0;
    int pos=-1;
    for (int var = 0; var < similitud.size(); ++var)
    {
        if (mayor_similitud<similitud[var].toInt())
        {
            mayor_similitud=similitud[var].toInt();
            pos=var;
        }
    }
    this->similitud=similitud;
    this->similitud.removeAt(pos);
    return QString::number(mayor_similitud);
}
```

Fig. 8: Estilo de codificación.

3.4 Pruebas de Software

El proceso de pruebas no es más que el proceso de ejecución de un programa con la intención de descubrir un error, una prueba tiene éxito si descubre un error no detectado hasta entonces. Las pruebas demuestran hasta qué punto las funciones del software parecen funcionar de acuerdo con las especificaciones. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad (Pressman, 2010).

La estrategia que se ha de seguir a la hora de evaluar dinámicamente un sistema software debe permitir comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Las pruebas unitarias están dirigidas a probar cada componente individualmente para asegurar que funcione de manera apropiada como unidad (Pressman, 2010).

3.4.1 Pruebas de Caja Blanca

Las pruebas de caja blanca se basan en un minucioso examen de los detalles procedimentales. Mediante las pruebas de caja blanca se obtienen casos de pruebas que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo; ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsas; ejecuten todos los bucles en sus límites y con sus límites operacionales; y ejerciten las estructuras internas de datos para asegurar su validez (Pressman, 2010). Para el *plugin* de los k vecinos más cercanos se utilizó la técnica ruta básica. La prueba de caja blanca le fue aplicada al método de Calcular_Similitud_Pearson.

Paso número 1:

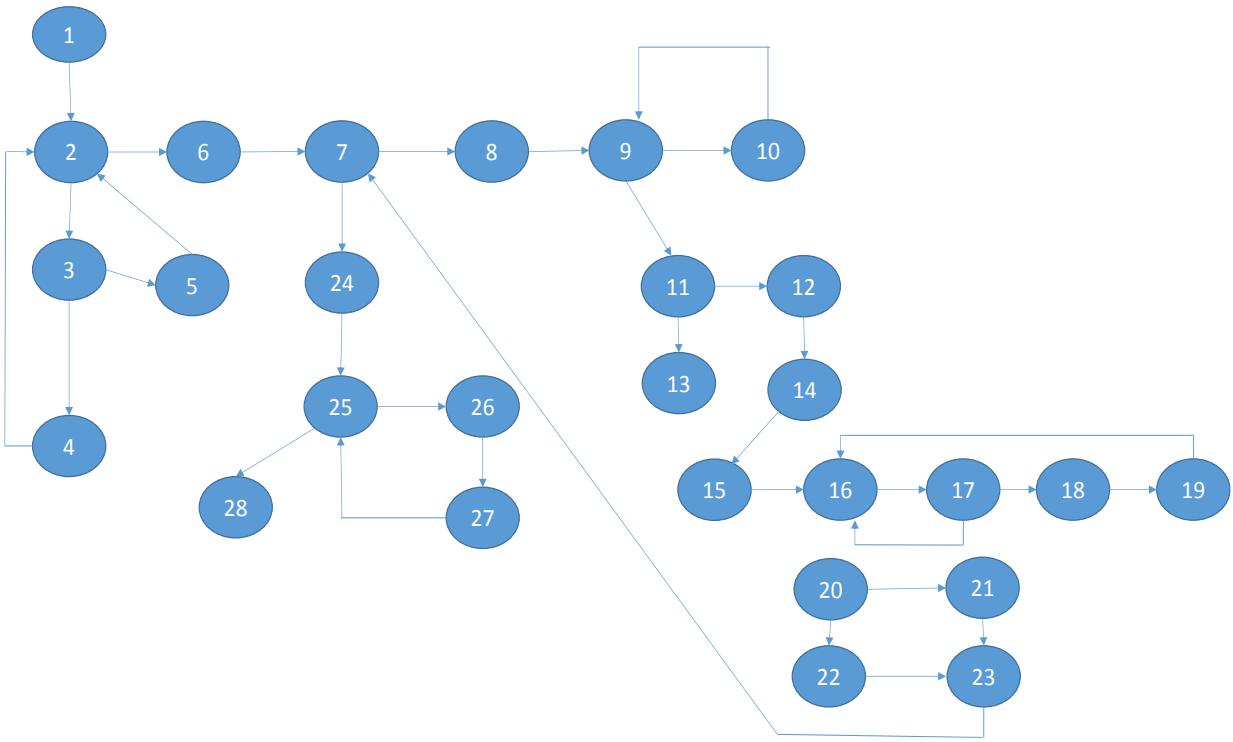

Fig. 9: Prueba de caja blanca.

Paso número 2:

Complejidad ciclomática: Aristas – Nodos + 2 = 37 - 28 + 2 = 11

Paso número 3:

Caminos Independientes:

C-1 = 1 – 2 – 6 – 7 – 24 – 25 – 28. (Insatisfactorio)

C-2 = 1 – 2 – 3 – 5 -2 – 6 – 7 – 24 – 25 – 28. (Insatisfactorio)

C-3 = 1 – 2 – 3 – 4 – 2 – 6 – 7 – 24 – 25 – 28. (Insatisfactorio)

C-4 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 24 – 25 – 26 – 7 – 25 – 28. (Insatisfactorio)

C-5 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 18 – 19 – 16 – 20 – 21 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28. (Insatisfactorio)

C-6 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 16 – 20 – 21 – 23 – 7 – 24 – 25 – 26 – 27 – 25 - 28. (Insatisfactorio)

C-7 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 18 – 19 – 16 – 20 – 22 – 23 -7 – 24 – 25 – 26 – 27 – 25 – 28. (Satisfactorio)

C-8 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 13 – 14 – 15 – 16 – 17 – 18 – 19 – 16 – 20 – 22 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28. (Satisfactorio)

C-9 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 16 – 20 – 21 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28. (Satisfactorio)

C-10 = 1 – 2 – 3 – 5 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 16 – 20 – 22 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28. (Insatisfactorio)

C-11 = 1 – 2 – 3 – 4 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 16 – 20 – 21 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28. (Satisfactorio)

Paso número 4:

Tabla 4: Tabla de descripción de la prueba de caja blanca.

Camino	Descripción	Resultado de la Prueba
1 – 2 – 6 – 7 – 24 – 25 – 28.	En el nodo uno se realiza la consulta a la base de datos con el fin de obtener los usuarios, además se crean e inicializan las listas usuarios _analizados y am_totales. En el paso dos se crea el while.	Insatisfactorio: La prueba resultó insatisfactoria ya que del nodo 2 se pasa para el nodo 6 algo que de acuerdo a la lógica de negocio y lógica de programación es imposible.
1 – 2 – 3 – 4 – 2 – 6 – 7 – 8 – 9 – 10 – 9 – 11 – 12 – 14 – 15 – 16 – 17 – 16 – 20 – 21 – 23 – 7 – 24 – 25 – 26 – 27 – 25 – 28.	En el nodo uno se realiza la consulta a la base de datos con el fin de obtener los usuarios, además se crean e inicializan las listas usuarios _analizados y am_totales. En el paso dos se crean el while con el fin de recorrer todos los usuarios e ir ubicándolos en la lista. En el nodo tres se realizan la comparación para saber si es el usuario activo y de ser así retorna para el nodo 2 para continuar con el while. En el nodo seis se crea un for para recorrer todos los usuarios analizados y calcular el coeficiente de correlación de Pearson, acción que se extiende por todos los nodos hasta llegar al nodo	Satisfactorio: La prueba resultó satisfactoria ya recorrieron todos los nodos del camino y la respuesta fue la deseada.

	veinticuatro donde se inicializa la lista similitud_copia que después en los siguiente nodos se le van adicionando valores hasta llegar el nodo veintiocho donde se retorna la lista y se acaba el método.	

3.4.2 Pruebas de Integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras al mismo tiempo, se lleva a cabo pruebas para detectar errores asociados con la interacción. Se clasifican en integración incremental (ascendente o descendente) e integración no incremental. La primera se combina el módulo que se debe probar con el conjunto de módulos que ya han sido probados, mientras que la segunda se prueba cada módulo por separado y luego se integran todos a la vez probando al sistema completo (Oterino, 2014).

En el componente para el cálculo de los k vecinos más cercanos se aplicaron las pruebas de integración incremental ascendente pues a medida en que se implementaba o modificaba algún componente se iban probando.

Para la realización de las pruebas, se tuvo que implementar una aplicación que simulara el trabajo del SGPM ya que el mismo no está utilizable. Luego se mandaron a generar recomendaciones pero esta vez usando el *plugin* desarrollado y el *plugin* de recomendación por filtrado colaborativo existente en la plataforma.

Debido a que el *plugin* de recomendación por filtrado colaborativo existente utiliza una cantidad fija de k-vecinos, se hace necesario implementar en dicho *plugin* un método en la clase principal llamado set_k, el cual permite que el valor de k sea variable y no estático. Además fue necesario agregar esta funcionalidad, ya que sin esta modificación el gestor no podría interactuar con el *plugin* de recomendación usando el valor que arroja el componente para el cálculo de los k vecinos más cercanos en la técnica de filtrado colaborativo.

El método mencionado persigue como objetivo general facilitar las pruebas, proceso que es complicado en el componente actualmente, debido a la manera en que se encuentra implementado, dado que usa un valor estático y los valores que se van a generar a partir del *plugin* desarrollado serán variables, lo que traería consigo recompilar el *plugin* tantas veces como pruebas se hagan. De esta manera se compila una sola vez y se accede a esa funcionalidad set_k anteriormente mencionada, para realizar las pruebas por cada valor

de k que se genere. El otro fin que persigue dicho método es dejar sentadas las bases para una futura comunicación entre el *plugin* de recomendación por filtrado colaborativo y el desarrollado en este trabajo de diploma, de manera tal que el SGPM recoja el valor que proporciona el componente que se desarrolló y lo entregue al *plugin* de recomendación por filtrado colaborativo.

3.4.3 Pruebas de carga y estrés

El *plugin* desarrollado debe desplegarse junto con el sistema de recomendación por filtrado colaborativo en entornos donde la cantidad de usuarios sea variable, por tal razón se realiza este tipo de prueba al *plugin*, cuyo objetivo es verificar y validar el desempeño de un elemento de un sistema bajo diferentes condiciones de carga, además estas pruebas son importantes cuando los sistemas deberán soportar un gran volumen de usuarios o transacciones concurrentes. Se debe tener en cuenta que estas pruebas deben ser realizadas bajo condiciones controladas para asegurar la precisión de las medidas tomadas. Por tal razones estas pruebas se realizaron entornos diferentes y con cantidades de usuarios variables, los resultados de las mismas se muestran a continuación en la Tabla 5:

Tabla 5: Pruebas de carga y estrés.

Cantidad de usuarios analizados	Propiedades de la PC	Tiempo de Respuesta(milisegundos)	Sistema Operativo	HDD(GB)
30	Intel Core i3 a 1.7GHz, 4Gb RAM.	8	Windows 8.1	460
500	Intel Core i3 a 1.7GHz, 4Gb RAM.	28	Windows 8.1	460
1000	Intel Core i3 a 1.7GHz, 4Gb RAM.	78	Windows 8.1	460
30	Intel Core i3 a 2.0GHz, 2Gb RAM.	6	Linux, Ubuntu 12.04	160
500	Intel Core i3 a 2.0GHz, 2Gb RAM.	27	Linux, Ubuntu 12.04	160

1000	Intel Core i3 a 2.0GHz, 2Gb RAM.	74	Linux, Ubuntu 12.04	160

Significado de los elementos de la tabla 5:

1. **Cantidad de usuarios analizados**: Hace referencia a la cantidad de usuarios que se analizaron.
2. **Propiedades de la PC**: Muestra las propiedades de las PC donde se ejecutó el *plugin* desarrollado.
3. **Tiempo de Retorno**: Tiempo que demora el *plugin* en generar la cantidad de k pertinentes en cada caso.
4. **Sistema Operativo**: Sistema Operativo de la PC donde se ejecutó el *plugin*.
5. **HDD (GB):** Capacidad del disco duro de la PC donde se ejecutó el *plugin*.

Después de realizadas las pruebas de carga y estrés, se arriba a la conclusión de que en el *plugin* desarrollado, varía considerablemente el tiempo de respuesta de las recomendaciones, en dependencia de la cantidad de usuarios que se analicen y de las propiedades que tenga la PC donde se ejecute el *plugin*. Como se puede observar en la Tabla 5, los tiempos de retorno no varían a gran escala, cuando se analizan la misma cantidad de usuarios. Esto se debe a que las maquinas donde se probó el *plugin* son similares.

3.4.4 Validación del componente para el cálculo de los k vecinos más cercanos

Las pruebas de validación verifican que el sistema de software producido cumple con las especificaciones y que logra su cometido (Pressman, 2010).

Para validar que el componente para el cálculo de los k vecinos más cercanos cumple con su cometido, se probó el *plugin* desarrollado junto con el *plugin* de filtrado colaborativo para diferentes cantidades de usuarios y diferentes k generados, así se pudo comparar la calidad de la recomendación así como el tiempo de respuesta.

Para esto se utilizó como métrica de evaluación el error absoluto medio (en adelante MAE – Mean Absolute Error). Esta medida mide la desviación media entre el valor de predicción de un elemento y el valor de puntuación real que asigna el usuario al elemento. Está considerada como una medida estadística para la estimación de la exactitud con la que el sistema realizará las predicciones. Utilizando esta métrica, cada elemento del conjunto de prueba es tratado de la misma manera. Esto significa que se le da el mismo peso

al error cometido en calcular la predicción de un elemento donde su valor real está por encima o por debajo del promedio de la escala de valores. El MAE se puede obtener de la siguiente manera (Herlocker, 2004):

$$MAE = \frac{\sum_{i=1}^{n} \boldsymbol{p}(\boldsymbol{u}, \boldsymbol{i}) - r_i}{P}$$

Fig. 10: Error Absoluto Medio.

Considerando P el número total de predicciones realizadas, $\boldsymbol{p}$ ($\boldsymbol{u}$,) el valor de la predicción para una calificación y ri el valor real de puntuación. De esta manera se calcula la desviación de las recomendaciones predichas con los valores reales, donde se considera que a menor MAE mejor predicción del sistema de recomendación (Paula Rodríguez, 2013).

Tabla 6: Resultados de la validación.

K Que Arroja	Base de Datos (usuarios)	Tiempo de Retorno (milisegundos)	MAE
5	20	8630	0.0883
3	20	10850	0.0669
5	500	28342	0.3238
7	500	31784	0.2709
5	1000	37659	0.4850
15	1000	40187	0.4267

Como se puede observar en la tabla anterior, mientras mayor es la cantidad de usuarios en la base de datos, mayor es el tiempo de respuesta. Se comprobó que a medida que varió el k en las diferentes bases de datos, fue disminuyendo el valor de sus correspondientes MAE, garantizando de esta manera una calidad aceptada en la predicción de las recomendaciones, en comparación con los valores arrojados mientras el k

era estático con valor igual a cinco. Por lo que no llega a ser representativo el aumento en el tiempo de respuesta, como si lo es la disminución en el MAE.

El algoritmo del *plugin* desarrollado tiene una complejidad temporal de $O(n^2)$ lo que permite tener un tiempo de respuesta controlado en cualquier entorno. Por otra parte la media de las diferencias de los MAE del *plugin* desarrollado y el existente es 0.0442, por lo que se arriba a la conclusión de que con esa complejidad temporal y esa media de variación del MAE se logra un balance entre el tiempo de respuesta de la recomendación y la calidad de la misma.

3.5 Conclusiones parciales

Al término de este capítulo se concluye que los estándares de codificación utilizados en el desarrollo del *plugin*, permiten un mejor entendimiento y facilidades de interpretación para su futura revisión por parte de desarrolladores, debido a la legibilidad y el estilo de codificación que presenta dicho *plugin*. La realización de la prueba de carga y estrés demostró el comportamiento del *plugin* en diferentes entornos, ya sea con la variabilidad de la cantidad de usuarios como también con la de las características del sistema, en ambos casos arrojó resultados diferentes. Con la realización de la prueba de caja blanca se concluye que se ejecutan al menos una vez todos los caminos independientes del *plugin*, se utilizan las decisiones en su parte verdadera y en su parte falsa y además se ejecutan todos los bucles en sus límites. Se obtuvo caminos insatisfactorios, lo que pudiera significar que existen errores en la lógica del programador o que el código fuente presenta errores en su estructura. Sin embargo después de haber realizado un análisis se comprobó el correcto funcionamiento del *plugin* así como la correcta estructura de su código fuente. Para concluir se hizo una prueba de validación en función de una variabilidad en el k que muestra, la cual arrojó como resultado que se logró un balance equitativo entre la calidad de una recomendación y su tiempo de respuesta.

Conclusiones

Al concluir el presente trabajo se llega a la conclusión que se obtuvieron todos los conocimientos necesarios para desarrollar un componente para el cálculo de los k vecinos más cercanos en la técnica de filtrado colaborativo en la plataforma AGORAV, el cual fue desarrollado con el objetivo de propiciar una cantidad variable de k, que se ajuste a la cantidad de usuarios existentes y a la variabilidad de sus preferencias. Una vez que se realizaron todas las tareas de investigación se puede afirmar que se cumplieron de manera satisfactoria, llegando a las siguientes conclusiones:

- Las herramientas y tecnologías utilizadas permitieron llevar a cabo de manera satisfactoria la implementación del componente correspondiente a la solución de la problemática inicial, así como facilitar su integración a la plataforma. El marco teórico de la investigación permitió sentar las bases científicas para llevar a cabo un estudio acerca de los conceptos asociados a las diferentes técnicas en el cálculo de los k vecinos más cercanos haciendo uso de la técnica de filtrado colaborativo.
- El uso de la metodología AUP brindó la posibilidad de generar solamente la documentación necesaria para la investigación, que pudiera ser entendida por otros desarrolladores para trabajar en versiones avanzadas del *plugin*.
- La utilización del patrón arquitectónico N-Capas permitió dividir el componente en tres capaz: Acceso a datos, Lógica de Negocio y Presentación, permitiendo el desarrollo del *plugin* en varios niveles, y en caso de que sobreviniera algún cambio, solo se atacaría al nivel requerido sin tener que revisar entre código mezclado.
- Los requisitos funcionales y no funcionales permitieron definir las características y capacidades que debía cumplir el *plugin*.
- Se le incorporó un método adicional al *plugin* de recomendación por filtrado colaborativo, el cual facilitó el proceso de pruebas a partir de los k variables que arroja como resultado el componente desarrollado. Se logró sentar las bases para una futura comunicación entre ambos *plugin* y el SGPM.
- Se logró desarrollar un *plugin* capaz de proporcionar un k variable, arrojando como resultado, una vez que el *plugin* de recomendación por filtrado colaborativo comenzó a usar los valores variables de k, un balance entre el tiempo de respuesta de la recomendación y la calidad de la misma en entornos sumamente cambiantes.

Recomendaciones.

Después haber culminado el presente trabajo y ver su resultado se propone la siguiente recomendación:

- Modificar el SGPM de manera tal que se emplee el componente para el cálculo de los k vecinos más cercanos, es decir que sea capaz de ejecutarlo, y a partir del resultado que arroje, proporcionarle el valor de k al *plugin* de recomendación por filtrado colaborativo existente, logrando así una interacción entre ambos componentes con sus nuevas modificaciones, y el SGPM.
- Integrar el *plugin* desarrollado al sistema de recomendación por filtrado colaborativo para así lograr una mayor rapidez a la hora de generar una recomendación a un usuario determinado.

Bibliografía

7Graus. 2015. Significados.com. [En línea] 2015. [Citado el: 10 de 06 de 2015.] http://www.significados.com/feedback/.

Alegsa, Leandro. 2010. Diccionario de Informática y Tecnología. [En línea] 2010.

Álvarez, Sara. 2012. desarrolloweb.com. [En línea] 2012. http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html..

Balabanovic, M. y Shoman, y,. 1997. *¨Content-based, collaborative recomendation¨.* 1997.

Burke, R. 2002. *Hybrid recommender systems: Survey and experiments. User.* 2002.

Carlos Kicillof, NIcolás y Reynoso. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Castells, Manuel. 2001. *Internet y la sociedad Red.* Cataluña : s.n., 2001.

Cedeño, Elvsi Ferrera. 2011. *Sistema de Recomendanción Basado en las preferencias de los usuarios para la plataforma PTARTV.* 2011.

Ciencias, Informáticas Universidad de las. 2008-2009. *Arquitectura y Patrones de diseño.* 2008-2009.

2012. Comunidad Digital de Conocimiento. *Comunidad Digital de Conocimiento.* [En línea] 25 de 09 de 2012. http://www.academica.mx/blogs/las-pruebas-integraci%C3%B3n-software.

Enrique Herrera-Viedma, Carlos Porcel,Lorenzo Hidalgo. 2004. Hipertext.net. [En línea] 2004. http://www.upf.edu/hipertextnet/numero-2/recomendacion.html.

Farfán, Gunther y Arias, Jesús. Scridb. http://es.scribd.com/doc/61498499/25trabajode-Scridb-Visual-Basic-2. [En línea]

Gonzalez, Rolando Alfredo Hernández León y Sayda Coello. 2011. *El Proceso de Investigación Científica.* Ciudad de la Habana : Editorial Universitaria, 2011.

Guo, X. 2006. *Personalized Government Online Services with Recommendation.* Sydney : s.n., 2006.

Herlocker, Jonathan L., y otros. 2004. *"Evaluating Collaborative Filtering Recommender Systems". New York : Journal.* New York : s.n., 2004.

Bibliografía

Hernández, Pier Paolo Guillen. 2006. [En línea] 06 de 06 de 2006. [Citado el: 11 de 06 de 2015.] http://pier.guillen.com.mx/algorithms/07-geometricos/07.8-centroide.htm.

Jacobson, Ivar,Booch,Grady y Rumbaugh. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Jiménez, Eduardo David Collado. 2013. *Base de Datos Única para la gestión de las.* 2013.

Larman, Craig. 2003. *UML y Patrones. 2da Edicion.* 2003. 2.

—. 1999. *UML y Patrones.* 1999.

Limonta, Mairena Arzuaga. 2014. *Plugin de filtrado colaborativo para generar recomendaciones con buena calidad en la plataforma VideoWeb 2.0 sin afectar el rendimiento del servidor web .* 2014.

Lobo, María Elena. 2005. *Curso para Aprender Programación.* 2005.

Lockhart, Thomas. 2009. *Tutorial de POstgres.* 2009.

Mondaray, Sergio Garcia. 2012. GodTic un mundo binario. [En línea] 15 de 11 de 2012.

Nieto, Sergio Manuel Galán. 2007. *Filtrado Colaborativo y Sistemas de Recomendación.* Madrid : s.n., 2007. pág. 8.

Oterino, Ana M. del Carmen García. 2014. javiergarzas. [En línea] 04 de 07 de 2014. http://www.javiergarzas.com/2014/07/tipos-de-pruebas-10-min.html.

Palacio, Juan. 2008. *ScrumManager: Gestión de Proyectos.* 2008.

Paula Rodríguez, Néstor Duque Méndez,Demetrio Ovalle,Valentina Tabares. 2013. *Validación de un sistema inteligente de recomendación híbrida en federaciones deI.* 2013. 3.

Pérez, Alexander Jesús Brown. 2012. *Sistema de Recomendación Basado en Contenido para la plataforma VideoWeb.* 2012.

Pressman, Roger S. 2010. *Software engineering, a practitioner·s approach.* New york : s.n., 2010.

Quiroga, Juan Pablo. 2010. *Requerimientos Funcionales y no Funcionales.* Colombia : s.n., 2010.

Ramos, Jesús. 2011. *Sistemas gestores de bases de datos.* 2011.

2011. Revisiones de código y estándares de codificación. [En línea] 2011. http://msdn.microsoft.com/eses/.

Bibliografía

Rodríguez, Antonio Pedro Albín. 2008. Sistema de Recomendación Colaborativo basado en algoritmos de filtrado mejorados. 2008.

Sarwar, B. 2000. *Analysis of recommendation algorithms for e-commerce.* 2000.

Sierra, Maria. 2008. *Trabajando con Visual Paradigm for UML.* 2008.

Sommerville, Ian. 2005. *Ingeniería de Software.* Madrid : s.n., 2005. séptima edición.

Torres, Emilio José Castellano. 2007. *Evaluación del uso de algoritmos colaborativos para orientar académicamente al alumnado en bachillerato.* Jaén : s.n., 2007.

Torres, Jesús. 2014. Proyecto Qt. Framework de desarrollo de aplicaciones. [En línea] 2014.

Viñolo, Raudel Raúl y Roquero, Alexander. 2012. *Sistema Gestor de Procesos de Media.* La Habana : s.n., 2012. v2.

2008. Visula Paradigm International. [En línea] 2008. [Citado el: 18 de 2 de 2015.] http://www.visual-paradigm.com/product/vpuml/features/..

Wiki, Qt. 2015. Qt Wiki. [En línea] 24 de 05 de 2015. https://wiki.qt.io/Category:Tools::QtCreator_Spanish.

Anexos.

1- Entrevista Realizada

Fecha: Enero 2014.

Entrevistadores: Randy Ruvira Castillo, Karell González Caro.

Entrevistados: Ing. Yarisel Rojas Castellanos (Líder del Proyecto CPM), Ing. Miguel Morciego Varona.

Preguntas:

1- ¿Qué beneficio aportaría para la plataforma AGORAV un *plugin* que permita calcular los **k** vecinos más cercanos para el *plugin* de recomendación por filtrado colaborativo?

2- ¿Cuáles son los requisitos fundamentales que debería cumplir el *plugin* a desarrollar?

Respuestas:

Pregunta 1: Mediante las respuestas obtenidas los dos entrevistados coincidieron que el beneficio más importante que este *plugin* aportaría a la plataforma AGORAV seria que así el *plugin* de recomendación se podría desplegar en cualquier ambiente de trabajo sin preocuparse cuan variable sean los mismos.

Pregunta 2: Fueron muchos los requisitos mencionados inicialmente, pero después de un profundo análisis los dos coincidieron en varios requisitos los cuales son:

- Garantizar que el *plugin* sea integrable con el SGPM.
- Buscar usuario de mayor similitud al usuario activo.
- Buscar el promedio de similitudes.