UNAD Universidad Nacional Abierta y a Distancia	**INFORME FINAL DE TRABAJO DE GRADO** *MUDABA QR*

MUDABA QM (MULTI DATABASE QUERY MANAGER)
TRABAJO DE GRADO

POR:

NELSON ANDRÉS ÁLVAREZ.

NOEL ANDRÉS ZARTA.

Trabajo realizado como requisito para optar al título de
Tecnólogo en Desarrollo de Software

ASESOR:
JOSÉ YESID AGUIRRE SÁNCHEZ

JURADO CALIFICADOR:
CARMEN EMILIA RUBIO VANEGAS

UNIVERSIDAD NACIONAL ABIERTA Y A DISTANCIA - UNAD
ESCUELA DE CIENCIAS BÁSICAS TECNOLOGÍAS E INGENIERÍA (ECBTI)
PROGRAMA TECNOLOGÍA EN DESARROLLO DE SOFTWARE
MAYO DE 2017

TABLA DE CONTENIDO

LISTA DE TABLAS

LISTA DE IMÁGENES

INTRODUCCIÓN

Las necesidades de manejo de información de las organizaciones son cambiantes y evolucionan a la par de las dinámicas digitales. Un software que cumple con las necesidades corporativas el día de hoy, puede no hacerlo el día de mañana. Es por esto que los diferentes productos de software utilizados deben ser continuamente actualizados o reemplazados por soluciones que cumplan con los fines técnicos y administrativos. Es menester de los desarrolladores disponer el uso de los sistemas desarrollados de tal manera que estos sean adaptables a las características de cada empresa y, asimismo, cuenten con una vida útil de eficiencia duradera.

Mudaba QM (Multi Database Query Manager) permite ejecutar simultáneamente consultas SQL que se distribuyen a través de la red hacia diferentes servidores interconectados, indicando el resultado de cada ejecución y permitiendo guardar las consultas en un archivo CSV localmente desde el cliente en el que se ejecuta. Dichos servidores, o terminales, pueden ser activados o desactivados desde el programa según las necesidades

Como características principales tenemos la capacidad de autocompletar las Queries. Esto es, consultas que se almacenan previamente en un archivo XML pueden ser accedidas escribiendo en el campo indicado las primeras letras de ésta. Lo anterior con el fin de no tener que repetir la escritura de aquellas Queries que se utilizan frecuentemente.

Mudaba QM facilita, además, la ejecución de conteos (SELECT COUNT) pues realiza una sumatoria general de los conteos individuales de cada terminal afectada, presentando una cifra consolidada del resultado.

Además de lo anterior, *Mudaba QM* incluye las principales funcionalidades de la consola de ejecución de un Sistema Gestor de Bases de Datos, como son: identificación de errores de conexión, sintaxis y construcción de la Query.

1. DEFINICIÓN DEL PROBLEMA

1.1. Línea de Investigación

Ingeniería de Software

Desarrollar experiencias de orden formativo y disciplinar en el campo de la investigación, con base a la construcción de software de forma sistémica y estructurada de acuerdo a los principios propios de la ingeniera de software.

1.2. Descripción del problema

En el ejercicio de su función, las empresas que manejan una gran cantidad de datos enfrentan retos de diversos tipos. Particularmente, existen necesidades de administración de datos que requieren la actualización de información en diferentes terminales o servidores locales conectados a la red empresarial. Cuando se requieren ejecutar consultas en diferentes servidores desde un Sistema Gestor de Bases de Datos, es necesario realizar dicha ejecución servidor por servidor; lo que resta eficiencia al proceso, máxime cuando no se cuenta con sistemas costosos de gestión de información o con software aplicado para cada función requerida. *Mudaba QM* facilita dicha gestión, permitiendo realizar ejecución de Queries simultáneas en los servidores señalados.

Aunque en principio la idea es simple, las posibilidades de la propuesta son muy amplias. Ejemplos de implementación serían los siguientes: un Call Center que requiere consultar y actualizar información constantemente entre sus asesores ubicados en distintos terminales; una

empresa de ventas que requiere gestión de información consolidada; manejo de bases de datos en sistemas de información empresariales; entre muchas otras posibilidades.

1.3. Formulación del problema

¿Cómo mejorar la eficiencia en la ejecución de consultas SQL multiservidor a bajo costo?

1.4. Sistematización del problema

¿Cómo ha sido el comportamiento del flujo de información referente a consultas múltiples en bases de datos y servidores disponibles?

Operacionalmente, ¿de qué manera afecta en la actualidad la búsqueda de información en las múltiples bases de datos y servidores la productividad de la empresa?

¿Qué impacto ha tenido el desempeño actual de la administración de las múltiples bases de datos y servidores en las empresas objetivo?

¿Quién sería el personal encargado de hacer las consultas múltiples?

2. JUSTIFICACIÓN

Actualmente, existe una gran cantidad de Gestores de Bases de Datos; cada motor de bases de datos (Postgresql, MySQL, Oracle, etc.) tiene el propio. Cada uno de estos gestores tiene la posibilidad de configurar múltiples servidores y bases de datos y ejecutar consultas individualmente; pero, no existe en el mercado una solución de software que permita distribuir dichas consultas de forma simultánea a todos los servidores conectados. *Mudaba QM* permite esta función.

El mercado objetivo del proyecto son empresas de servicios de comunicaciones, mercadeo, call centers y, en general, todas aquellas empresas que manejen bases de datos sistematizadas y cuyo flujo de información esté cambiando constantemente. Además, el software es útil para aquellas organizaciones a las que por motivos económicos no les es posible implementar soluciones completas con aplicaciones cliente/servidor; convirtiéndose nuestra propuesta en una opción efectiva y de bajo costo para manejo de información en bases de datos.

Posiblemente una de las barreras principales para la oferta del producto es la dificultad de entrada a las organizaciones y la puesta a punto para que éste se adapte a las necesidades particulares de cada empresa. Una vez superados estos obstáculos, el panorama para la implementación se vuelve mucho más claro y se amplían las posibilidades del software. Por otro lado, la propuesta que se plantea no va dirigida al usuario común, sino a un usuario que tenga conocimientos sólidos de bases de datos y entienda la responsabilidad que implica ejecutar una consulta en una base de datos.

3. OBJETIVOS

3.1. Objetivo general

Desarrollar un software que reemplace en ciertas funciones a un Sistema Gestor de Bases de Datos tradicional, el cual permita una configuración sencilla de los distintos servidores para ejecución de consultas múltiples de forma rápida y simultánea.

3.2. Objetivos específicos

- Desarrollar una alternativa eficiente y multipropósito a los Sistemas Gestores de Bases de Datos tradicionales
- Agilizar la ejecución de consultas sobre bases de datos
- Ofrecer un sistema de consultas de bases de datos que pueda reemplazar soluciones más costosas y aplicadas para empresas que no tengan los recursos para adquirir soluciones particulares de gestión de información; Especialmente para procesos que requieren manejo de bases de datos homogéneas

4. DELIMITACIÓN

La propuesta está diseñada para que sea útil a una amplia variedad de medianas empresas. Desde empresas de servicios hasta empresas productoras de bienes. Esto es debido a que la penetración de la tecnología, en particular para el manejo de información, se ha extendido debido al bajo costo de la misma. Pero la implementación de las nuevas tecnologías trae consigo nuevas necesidades de gestión de la información, así como de conocimientos para soportarlas. Es así como, a pesar de que la utilidad potencial del software se extiende a un gran número y tipos de empresas, su uso requiere de cierta preparación y conocimientos previos en Bases de Datos Relacionales. Entonces, de entrada, la principal limitación es la barrera del conocimiento, la responsabilidad que requiere el manejo de bases de datos a través de sentencias SQL; conocimientos que no todo el mundo posee.

Otra limitación del software tiene que ver con el motor de bases de datos a utilizar. Ya que, aunque es posible hacer que el software permita conectar a los principales gestores de bases de datos como MySQL, Oracle, Postgresql y demás, por motivos de tiempo el desarrollo se enfocará en la conexión por medio de Postgresql, debido a que es la que se maneja en el caso empresarial aplicado.

5. MARCO DE REFERENCIA

5.1. Marco teórico

Todo el desarrollo de Mudaba QM estará basado en el paradigma de Programación Orientada a Objetos (POO) y en la administración de Bases de Datos Relacionales. Ambos conceptos se profundizarán más adelante en el marco conceptual; este espacio se utilizará para mencionar parte de la bibliografía en la que está basado el proyecto relacionada con estos temas.

Aunque de POO se viene hablando desde la década de 1960, fue en la segunda mitad de la década de 1990 cuando empezó a ser estudiada y aplicada eficientemente en los diferentes lenguajes existentes como C y C++, así como a los que apenas se estaban popularizando como Java y C#. Danforth y Tomlinson (1998), nos hablan de los beneficios de acoger el paradigma de POO, como la modularidad y la extensibilidad del código; así como de la correcta interacción que deben tener los objetos de un sistema. Se analizan varias teorías que representan los conceptos de POO, empezando por la forma en que se tratan los tipos de datos abstractos y la herencia entre clases.

Por su parte, Booch y Maksmchuk (2007), se enfocan en el análisis y diseño de POO, desde un punto de vista aplicado. Incorporan en su texto conceptos de UML (Unified Modeling Language), así como de algunos lenguajes orientados a objetos, como Java y .Net. Más allá, estudian diferentes métodos para el desarrollo OO y ofrecen múltiples ejemplos básicos y complejos mediante casos de estudio reales.

Paralelamente a la implementación de la POO, es importante relacionarnos con las técnicas de vanguardia para el desarrollo de sofware. Hoy en día, no es suficiente con entender la lógica, sino que también hay que codificar rápido y eficientemente, teniendo en cuenta que la mayor parte del tiempo se programa en equipos de trabajo y procurando generar código mantenible en el tiempo. Aquí entra *The Pragmatic Programmer*, en donde Hunt y Thomas (1999) nos dan una serie de consejos y técnicas para llevar a buen término cualquier proyecto de programación. Este libro siempre permanecerá vigente, debido a su sencillés y a la adaptabilidad de sus conceptos y consejos a las nuevas tecnologías y se convierte en un libro 'obligado' para todo aspirante a programador.

Ahora bien, todo programador, además de dominar uno o varios lenguajes para codificar un sofware, debe también entender los conceptos de *Bases de Datos Relacionales (BDR)*, que son inherentes a cualquier sistema de información y, por lo tanto, a la gran mayoría de proyectos importantes que emprendamos. Este conocimiento no se debe limitar a saber ejecutar las sentencias SQL necesarias; además, el desarrollador debe tener un buen entendimiento de la teoría de BDR, que le servirá principalmente para realizar diseños de sistemas más robustos y complejos. Para este proyecto, la teoría tratada en Date (2001), fue pertiente y completa para desarrollar los conceptos relacionados con bases de datos que la aplicación demanda.

Para terminar, Richardas (2015) nos orienta acerca de los diferentes patrones de arquitectura en el desarrollo de sofware. Particularmente importante para nuestro caso es el de Modelo Vista Controlador (MVC), que si bien no utilizaremos en toda su extensión durante el desarrollo del proyecto, sí nos servirá de base para diseñar la separación de las diferentes capas que componen la aplicación; esto es, la capa de interfaz, la capa de datos y la capa lógica.

5.2. Marco conceptual

Programación orientada a objetos. Cuando nos enfrentamos a un nuevo desarrollo, una de las primeras decisiones que se deben tomar a nivel metodológico es el tipo de modelo a aplicar. Programación estructurada, programación modular, programación funcional, programación lógica, programación Orientada a Objetos; son los paradigmas más populares de programación. El proyecto *Mudaba QM* será desarrollado con la metodología propuesta por la Programación Orientada a Objetos (POO), que favorece el desarrollo e implementación de sistemas complejos.

En la POO, los sistemas son construidos desde objetos auto-contenidos que interactúan entre ellos vía mensajes. Debido a la modularidad que permite, favorece el diseño, la implementación y el mantenimiento de dichos sistemas. Además, gracias a la posibilidad de herencia entre clases, favorece la reutilización de código y el desarrollo de sistemas extensibles.

El término “orientado a objetos” es usado para describir lenguajes de programación en donde los objetos de computación son (en cierta forma) como personas. En POO, los objetos generalmente tienen una identidad o naturaleza que persiste a través del tiempo independientemente de los cambios en el estado del objeto. Los objetos son inteligentes y responden a peticiones, o mensajes, dirigidas a ellos y que afectan al sistema. En términos del modelo, una respuesta del modelo al mensaje puede ser cambiar su estado interno, enviar mensajes a otros objetos, devolver una respuesta, crear nuevos objetos, o todo lo anterior. Una prescripción para manejar mensajes es comúnmente llamada *Método*.

Pero para hacer cálculos con objetos, estos deben existir primero, lo que esencialmente se puede hacer de dos formas: crearlos a partir de otros objetos prototipo o crearlos a partir de una *clase*, lo cual es el enfoque más común. Una clase puede ser pensada como una plantilla que identifica los métodos y *variables de instancia* a ser usados por el nuevo objeto para administrar los mensajes y almacenar los estados del objeto, respectivamente.

Dos conceptos fundamentales en la POO son Tipos de Dato Abstractos (TDA) y herencia. Los objetos encapsulan un estado, en conjunto con métodos para manejar dicho estado y de esta manera proveen la abstracción de los datos a través de la interfaz de los mensajes. Podría entonces un objeto verse y usarse de manera enteramente análoga a un TDA en los lenguajes estructurados tradicionales. Pero si los TDA son tan similares a los objetos, ¿acaso no es posible que la POO sea simplemente un modelo de programación en donde todos los datos son abstractos y toda la manipulación de datos es implementada vía operaciones TDA? Esta es una conjetura razonable. Sin embargo, la POO considera más aspectos que un mero soporte para los TDA; particularmente, *herencia.*

La principal idea de detrás del concepto de herencia es la ampliación del poder descriptivo respecto a la creación del objeto, durante la cual los métodos a ser usados por el nuevo objeto para manejar varios mensajes deben estar indicados de alguna forma. En lenguajes basados en clases, la herencia permite a los métodos usados por los objetos de una clase determinada, ser especificados de forma modular y extensible por medio de la ubicación lógica de métodos relacionados en clases individuales, relacionando dichas clases por medio de una jerarquía de subclases. De esta forma, los métodos potencialmente disponibles para un objeto de una clase

particular no son solo los métodos contenidos en la clase del objeto, sino también todos los de sus ancestros dentro de la jerarquía de la clase.

Teniendo en cuenta todo lo anterior, Mudaba QM, será desarrollado en lenguaje C#, perteneciente a la plataforma .Net de Microsoft. C# implementa el paradigma orientado a objetos incorporando la definición de clases, métodos y parámetros.

Bases de Datos Relacionales. El modelo relacional es el fundamento de la tecnología moderna de bases de datos, que hace de este campo una ciencia. Dicho modelo se encarga fundamentalmente de tres aspectos de la información: la estructura de datos, la manipulación de datos y la integridad de los datos. Aquí es importante identificar los términos estructurales más importantes para cada parte. Estos son *relación, tupla*, correspondiente a un registro de una tabla; *cardinalidad*, equivalente al número de tuplas; *atributo*, equivalente a una columna, *grado*, equivalente al número de atributos; *dominio*, conjunto de valores de donde se toman los valores de atributos específicos de relaciones específicas; y *clave primaria*, identificador único de cada tupla.

Finalmente, el modelo relacional no está completo sin el medio para operarlo. Este medio es el Lenguaje Estructurado de Queries (SQL).

5.3. Marco legal

[1]Con el auge del desarrollo de las aplicaciones informáticas se hace necesario la regulación y control en el desarrollo de software, todo esto dio lugar a la discusión sobre la mejor manera de proteger jurídicamente el componente intelectual involucrado en tales creaciones, debido a que los programas de computador (software) son una propiedad intangible difícil de clasificar.

Evidenciando la importancia del software para el desarrollo de la tecnología informática, autoridades nacionales e internacionales iniciaron debates para aclarar esta situación. Surgieron dos modos principales de protección de la propiedad intelectual: los derechos de autor y las patentes de invención.

A continuación, se presentará los mecanismos legales utilizados para proteger los programas de computador y, específicamente, con la posibilidad de protección por vía de patente. Para este efecto, se examinará primero el régimen de derechos de autor como la principal figura empleada para proteger el software a nivel mundial, considerando sus principales desarrollos en los Estados Unidos de América, Australia, Europa y Colombia.

- Protegiendo la expresión de una idea: la protección del software mediante derechos de autor.

[1] Basado en https://www.unisabana.edu.co/fileadmin/Documentos/Derecho/CEDEPI/Patentabilidad_del_Software.pdf

- Derechos de autor, copyright y la expresión de la idea: el primero en el sistema de derecho civil o sistema latino y el segundo en el sistema de *Common Law*[2]. El sistema latino se protege ante todo al autor mismo, a quien se le atribuyen dos clases básicas de prerrogativas, a saber, los derechos morales y los derechos patrimoniales. El sistema de *Copyright*, como su mismo nombre lo sugiere, hace énfasis en el derecho a copiar o a reproducir una determinada obra. La misma naturaleza patrimonial de este derecho implica que, generalmente, el titular de la obra sea un empresario (o una empresa) dedicado profesionalmente a la explotación económica de estos trabajos. El sistema de copyright entonces gira alrededor de la naturaleza económica de sus derechos como, por ejemplo, se protejan obras que no son suficientemente creativas, pero que si son el resultado de grandes esfuerzos empresariales y de la inversión de ingentes sumas de dinero: grandes bases de datos, obras multimedia, nuevas ediciones digitalizadas, o, como en el caso específico del proyecto desarrollado, en el desarrollo de software para la administración de múltiples bases de datos y servidores.

  En nuestro país, exponente del sistema latino de derechos de autor, ese principio se encuentra contemplado de manera particular en el artículo 6 de la Ley 23 de 1982 (Ley de Derechos de Autor) y en el artículo 7 de la norma andina sobre derechos de autor, la Decisión 351 de 1993.

[2] La Copyright Act (1976) de los Estados Unidos establece en la sección 102 (b) que “En ningún caso la protección del Copyright en un trabajo original se extiende a la idea, procedimiento, proceso, sistema, método de operación, concepto, principio o descubrimiento, cualquiera que sea la forma en que esté descrito, explicado, ilustrado o incluido en dicha obra.”

La Ley 23 de 1982, permitió que se incluyeran los programas de computador dentro del objeto de protección de nuestro régimen de derechos de autor. El artículo 2 de dicha ley establece que los derechos de autor se predican respecto de obras científicas, literarias y artísticas "cualquiera que sea el modo o forma de expresión y cualquiera que sea su destinación (…) que pueda reproducirse, o definirse por (…) cualquier otro medio conocido o por conocer". De todas maneras, aunque existía acuerdo general respecto de la aplicación de esta norma al software, se consideró necesario reglamentar lo relacionado con su inscripción en el Registro Nacional de Derecho de Autor, lo que se logró mediante el Decreto 1360 de 1989.

5.4. Hipótesis

Es posible mejorar la eficiencia en el manejo de la información de una organización con software de bajo costo para la administración de bases de datos.

6. ESTUDIO ECONÓMICO Y FINANCIERO

6.1. Determinación de inversiones y costos a partir de las variables técnicas

Para el desarrollo de Mudaba QM se realizó el avalúo de los computadores personales (PC) y la compra del *Switch* para configuración de la red local, descritos a continuación:

COSTOS DIRECTOS			
Materia Prima			
Materia prima	**Cantidad**	**Valor unitario**	**Valor total**
Pc Administrador	2	$ 2.100.000	$ 4.200.000
Switch Tp-link 16 Puertos	1	$ 120.000	$ 120.000
Costo total de materia prima			**$ 4.320.000**

Tabla 1. **Costos directos: materia prima**
Fuente: Elaboración propia

Para estimar el costo de las horas empleadas en el desarrollo del producto se tendrá en cuenta un precio promedio del valor actual de la hora/hombre de desarrollo. [3]*El costo promedio en el mundo por hora de programación es de 30.00 USD, en algunos países es más (Europa y Estados Unidos son alrededor de 120.00 USD), en algunos menos (México entre 20 y 30 USD) y en otros mucho menos (India 10 USD).*

[3] Tomado de http://objetopersistente.blogspot.com.co/2006/05/como-ponerle-precio-tu-software.html

Teniendo en cuenta esto, haremos una aproximación con el valor de 30 UDS (aprox. $87.000 COP) por hora. El tiempo destinado al desarrollo de Mudaba QM es de aproximadamente 6 meses, donde dos meses fueron destinados a la investigación (levantamiento de requisitos y requerimientos) y 4 meses de diseño y desarrollo (programación); en total se invirtieron 150 horas entre los dos desarrolladores. A continuación, se relaciona el costo promedio de la mano de obra como capital de trabajo.

COSTOS DIRECTOS			
Mano de obra			
Cargo	**Horas**	**Valor hora**	**Total / 6 meses**
Director de desarrollo	90	$ 87.000	$ 7.830.000
Auxiliar de desarrollo	60	$ 87.000	$ 5.220.000
Total, horas	120		
Costo total de mano de obra			**$ 13.050.000**

Tabla 2. **Costos directos: mano de obra**
Fuente: Elaboración propia

Al calcular los costos de materia prima y de mano de obra¸ determinamos que Mudaba QM tiene un costo aproximado de **$ 17.370.000** para su producción**.**

6.2. Estimación precio de venta

Determinados los costos en los que se incurre en el desarrollo de Mudaba QM, se procede a calcular un valor de venta al producto para recuperar la inversión en materia prima, mano de obra e investigación.

Como posible estrategia de venta, y teniendo en cuenta que es poco factible que un solo cliente pague $ 17.370.000 por el software; proyectamos que a un precio de $3.500.000 la inversión se recuperaría al acreditar la quinta venta. Si en el transcurso de los primeros 6 meses, el producto se vende a 10 clientes, duplicaríamos lo invertido, generando ganancias, y así sucesivamente si el producto se vende a más de 10 clientes. El precio tendrá algunas varianzas (como cantidad de puntos de red a configurar, tipo de base de datos a utilizar, etc.) dependiendo del tipo de empresa en donde se ponga en marcha, además también habrá costos de mantenimiento, sostenimiento y desarrollo del producto a lo largo del proceso de comercialización.

7. DISEÑO, IMPLEMENTACIÓN Y DOCUMENTACIÓN

7.1. Requisitos funcionales

- Ejecución de comandos SQL en bases de datos conectadas a la LAN
- Configuración de múltiples servidores y bases de datos
- Mostrar servidores activos y perfiles, y poderlos activar y desactivar. Manejar esto mediante *checkboxes*.
- Autocompletar. Permite guardar Queries personalizadas con un nombre determinado, de manera que cuando se escriban las primeras letras del nombre en el campo de la Query, se desplieguen sugerencias de Queries que coincidan con ese nombre para poder elegirlas y ejecutarlas.
- Sumatoria del count de todos los servidores. Con una simple orden se podrá contar los mensajes de todos los servidores señalados y que cumplan una condición.
- Guardar resultados de sentencias SELECT en archivo plano delimitado por comas (CSV), no debe guardar resultados vacíos.

7.2. Requisitos no funcionales

- Errores de sintaxis deben especificar el error
- Que se borre el campo de Texto al presionar un icono de limpieza
- En consultas múltiples, si algún servidor no puede conectar, la consulta debe mostrar el error y continuar

- La carpeta Outputs (en donde se guarden las consultas) debe abrir automáticamente después de un SELECT (siempre que existan resultados)
- Acceder a los archivos de confuguración y a carpeta de Outputs por medio del menú.

8. DISEÑO UML

8.1. Diagrama Casos de Uso

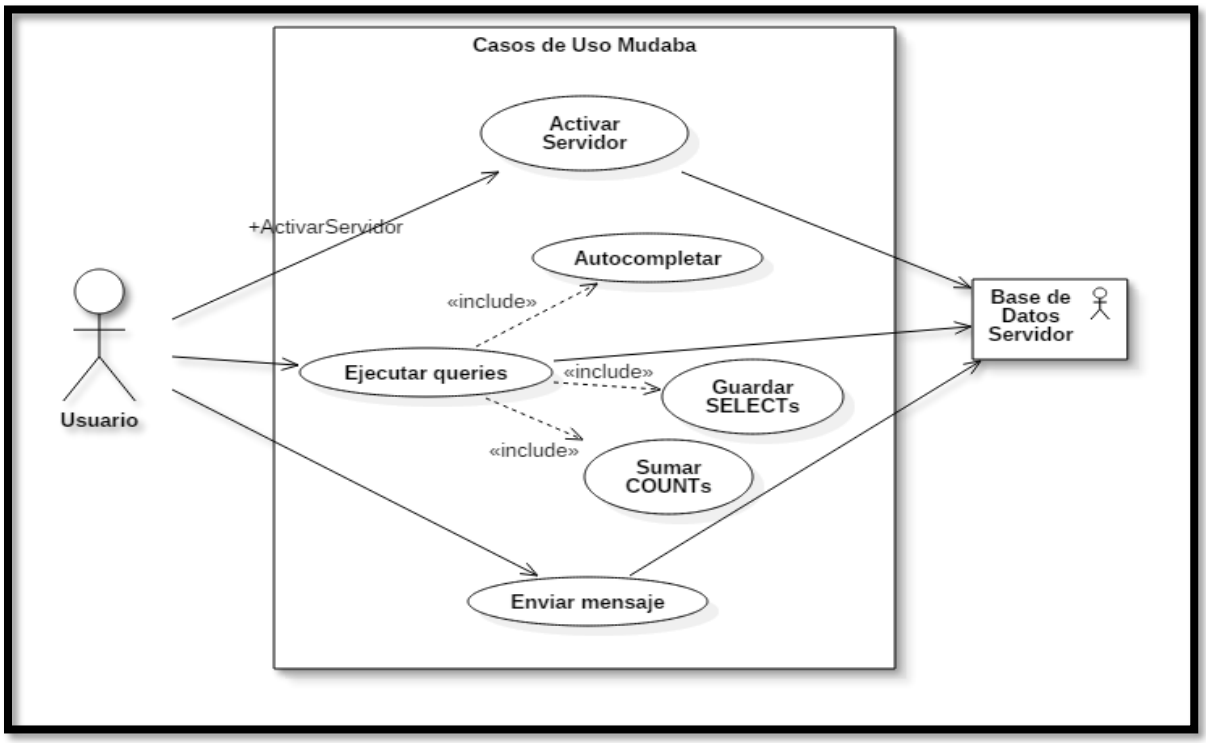


Imagen 1. **Diagrama casos de uso MudabaQM.**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qMmh2R19yYlN6QWM/view?usp=sharing

Escenario Casos de Uso. Mudaba QM permite ejecutar Queries que se distribuyen a través de la red local hacia los distintos SC conectados a la red, indicando el resultado de cada ejecución y permitiendo guardar las consultas en un archivo CSV. Antes de ejecutar la Query, el usuario activa los servidores que van a ser afectados. Luego, escribe un comando SQL en el

campo de texto para que se ejecute en los servidores seleccionados. Si el usuario empieza a escribir en el campo de autocompletar texto, se despliegan opciones para autocompletar lo que digita; el usuario elige una de ellas para ejecutarla. El resultado de las consultas SELECT son guardadas en un archivo .CSV. Al ejecutar una Query tipo SELECT COUNT, el sistema muestra el conteo de cada servidor, además de una sumatoria total de los resultados.

Título	Ejecutar Queries
Actor	Usuario
Escenario	El usuario escribe un comando SQL en el campo de texto para ejecutar en los servidores previamente definidos

Título	Autocompletar
Actor	Usuario
Escenario	El usuario empieza a escribir en el campo de texto y el sistema despliega opciones contenidas en un archivo XML para autocompletar lo que digita. El usuario elige una de ellas para ejecutarla

Título	Activar y desactivar servidores
Actor	Usuario
Escenario	Antes de ejecutar la Query, el usuario activa los servidores que van a ser afectados

Título	Guardar Selects
Actor	Usuario
Escenario	El sistema guarda el resultado de las consultas SELECT en un archivo .CSV

Título	Sumar COUNTs
Actor	Usuario
Escenario	Al ejecutar una Query tipo COUNT, el sistema muestra el conteo de cada servidor individualmente y, al final, una sumatoria total de los resultados

Tabla 3. Escenarios casos de uso
Fuente: Fuente: Elaboración propia

8.2. Diagrama de Objetos del Sistema

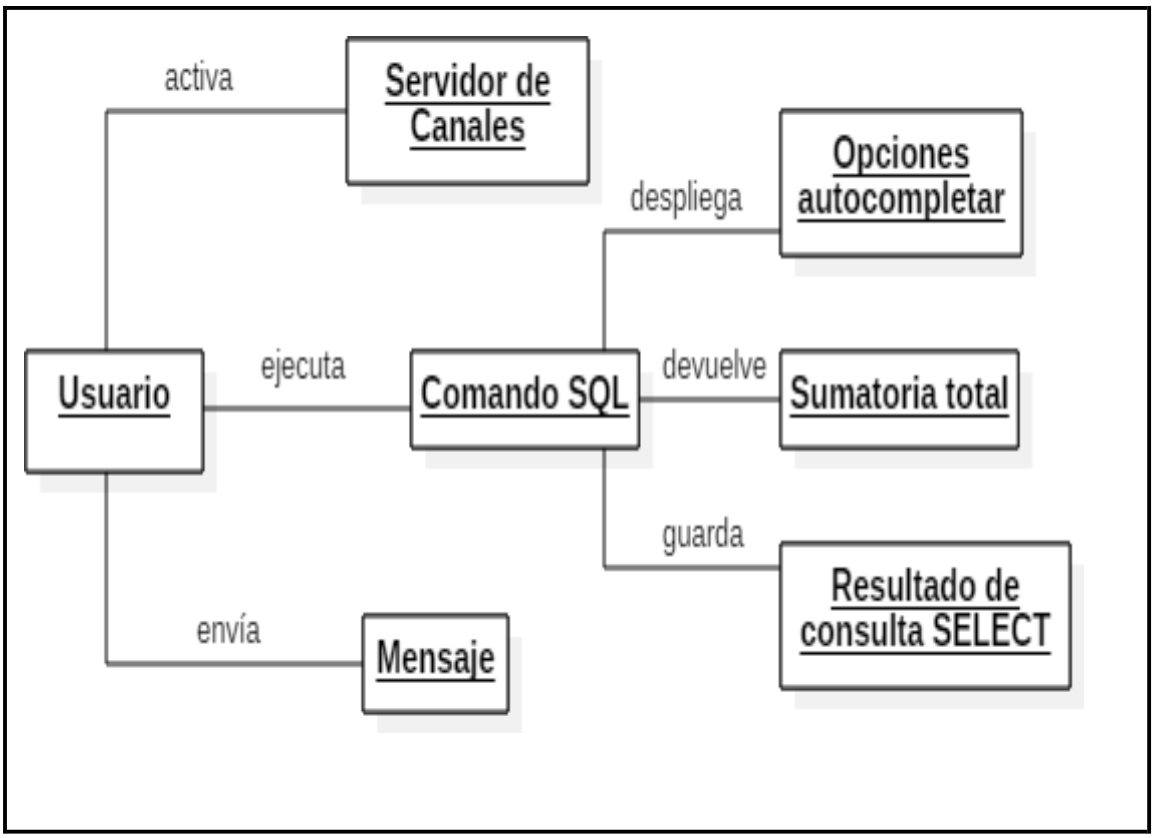


Imagen 2. **Diagrama de objetos del sistema MudabaQM**
Fuente: Elaboración propia. Recuperado de https://drive.google.com/file/d/0B2qtUbgdxp7qTjdoNXBRRzdJZ0k/view?usp=sharing

8.3. Diagrama de Clases

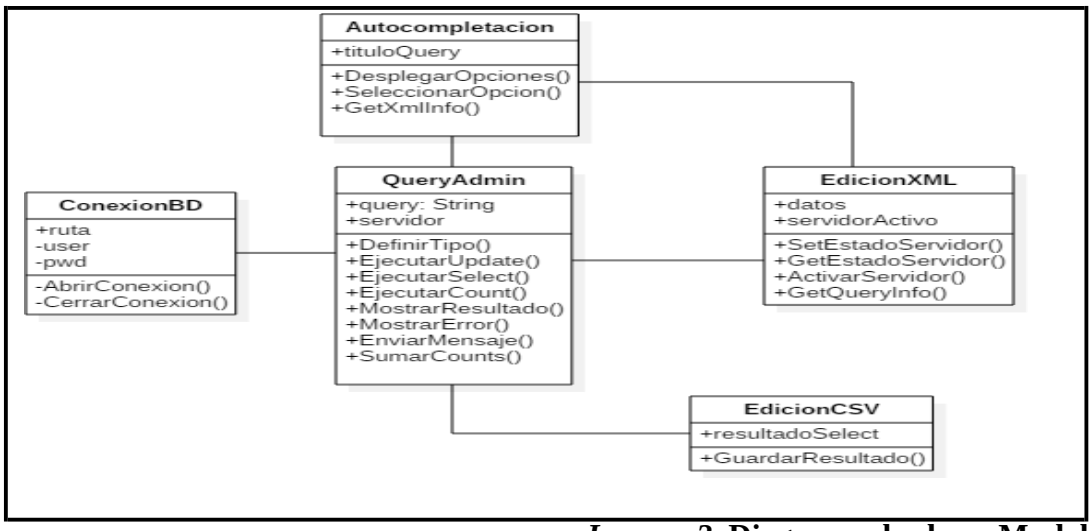


Imagen 3. **Diagrama de clases MudabaQM**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qRXNxMFpDMW1ZNlU/view?usp=sharing

8.4. Diagramas de Secuencia

- ***Activar servidor***

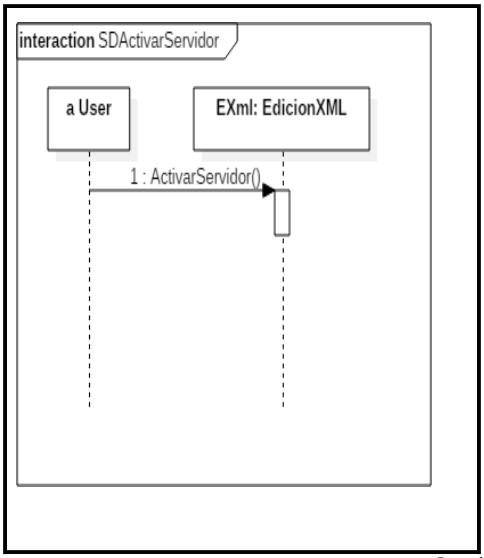


Imagen 4. **Activador servidor MudabaQM**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qcmtGNjdmbVR4MFU/view?usp=sharing

- ***Ejecutar Query***

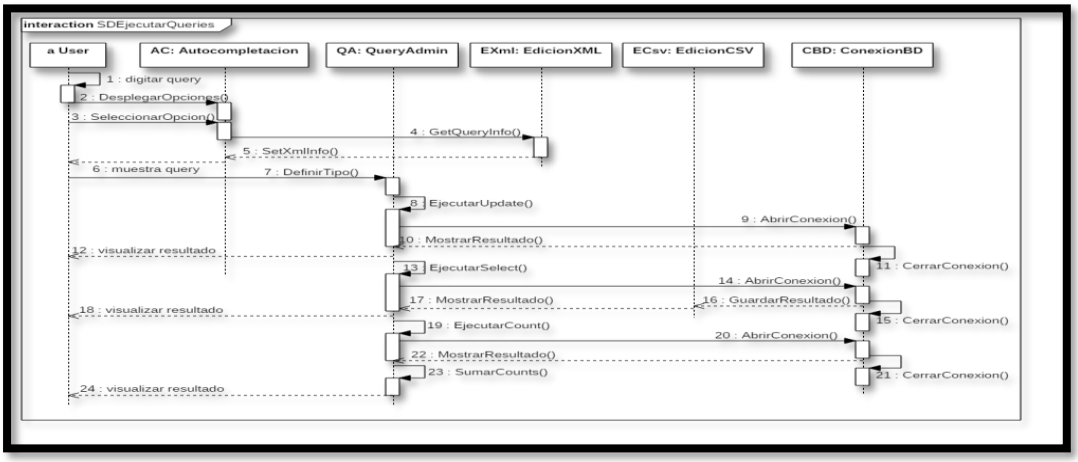


Imagen 5. **Ejecutar Query MudabaQM**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qUk82bVBCS2xqSGM/view?usp=sharing

8.5. Diseño de Datos

Mudaba QM es un gestor de bases de datos externas. Por esta razón, no cuenta con una base de datos propia sino con las rutinas necesarias para administrar otras bases de datos. Por lo tanto, el diseño de datos es especificado por el usuario por medio de las sentencias SQL que éste ejecute. No obstante, para efectos de configuración, Mudaba QM cuenta con un manifiesto para los parámetros de configuración nombrado *urls.xml*. Éste se encuentra ubicado en la carpeta principal del sistema una vez instalado y sirve establecer valores como los nombres a visualizar de los servidores, las IP y el estado (activo, no activo).

Por otro lado, existe también dentro del sistema una carpeta que almacena el resultado de las consultas SELECT que realiza el usuario. Esta carpeta *Outputs* guarda dicha información en archivos planos con valores separados con comas (archivo .CSV) para que el usuario pueda consultarla en cualquier momento.

8.6. Diseño de interfaz de usuario y ayudas

Descripción Ventana principal

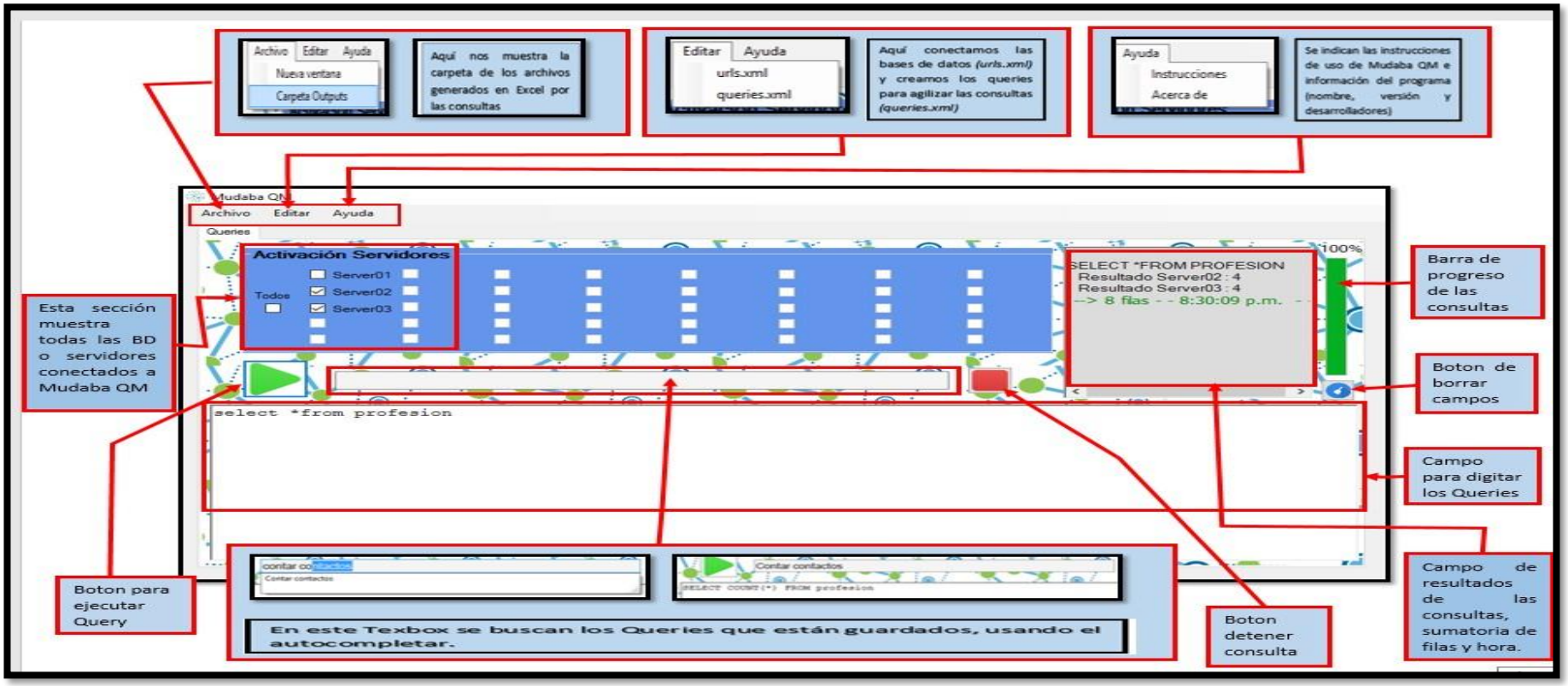


Imagen 6. **Descripción ventana principal MudabaQM**
Fuente: Elaboración propia. Recuperado de https://drive.google.com/file/d/0B2qtUbgdxp7qSXRoLXpMUWV0S2c/view?usp=sharing

Descripción interfaz:

Activación de servidores. En esta sección se registran los servidores que componen la red. Los servidores activados son aquellos sobre los que se correrá la Query, siempre y cuando tengan chequeada la casilla correspondiente. Como la cantidad de servidores varía según la organización en la que se instale el sistema, los espacios disponibles también variarán. Inicialmente, se plantea un sistema con espacios hasta para 40 servidores, que pueden ser ampliados o reducidos según el caso.

Texto autocompletar. Las Queries que se usan muy a menudo tienen la posibilidad de ser guardadas en un archivo XML, de manera que puedan ser accedidas fácilmente a través del campo autocompletar. En él, se escriben las primeras letras del nombre del archivo en el que se guardó la Query e inmediatamente se despliegan las opciones de archivos que coincidan con esas letras. El usuario elige el nombre de la *query* que busca y el contenido de ésta se mostrará en el campo de resultados.

Query. En este campo va la Query que se ejecutará. El usuario puede usar el campo de autocompletar para desplegarla o puede digitarla manualmente o puede copiarla y pegarla desde cualquier otro procesador de texto.

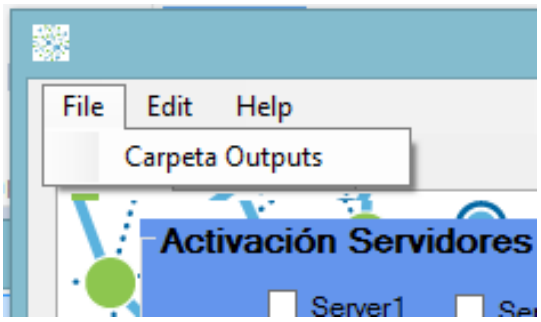
Campo de resultado. Aquí se muestra el resultado de las consultas efectuadas: el nombre del servidor y el número de registros afectados. También informa si hubo algún error con la sintaxis de la *query* o con la conexión a la base de datos. Además, realiza una sumatoria total de los resultados de todos los servidores afectados.

Botón ejecutar. Una vez la query está en el campo de escritura, se presiona este botón para ejecutarla. También puede presionarse la tecla de función F5.

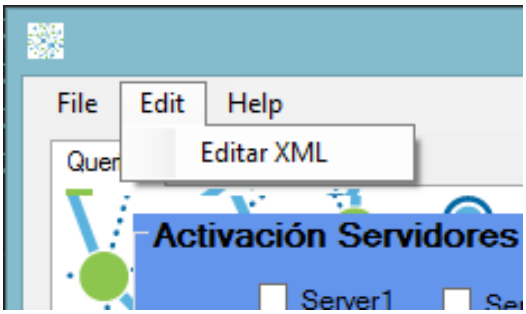
Limpiar. Borra toda la información de los campos

Menú. En esta sección se encuentran las ayudas del sistema.

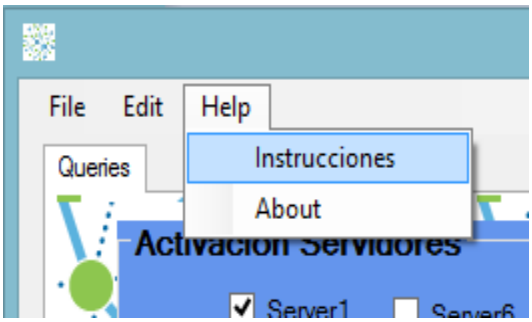
Ayudas. Por el momento el diseño de ayudas se limita a lo más básico, a medida que se avance en el desarrollo se irá complementando este menú.



La carpeta *Outputs* contiene los archivos .CSV resultado de las consultas tipo SELECT



Como se mencionaba anteriormente en este trabajo, el archivo XML hace las veces de manifiesto para la configuración del sistema.



Las instrucciones del sistema contienen los detalles de uso del software para los usuarios que no estén familiarizados con el mismo. Por su parte, la sección *Acerca de* contiene la información acerca de los autores.

Imagen 7. **Pantallazos menú MudabaQM versión 1.0.0**
Fuente: Elaboración propia

9. RESULTADOS DEL PROYECTO

9.1. Instrucciones de uso, configuraciones y ejecución de Mudaba QM versión 1.1.0

- Se deben tener los datos de las bases de datos para configurar la conexión con Mudaba QM (nombre, contraseñas, etc.) Esto se hace en el archivo *urls.xml*. Ejemplo:

urls.xml: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<?xml version="1.0" encoding="utf-8" ?>
<urls>
  <sl>
    <name>Server01</name>
    <estado>1</estado>
    <checkable>yes</checkable>
    <url>localhost</url>
    <puerto>5432</puerto>
    <db>ServerCompras</db>
    <user>postgres</user>
    <pwd>andreszan</pwd>
    <timeout>3</timeout>
  </sl>
  <sl>
    <name>Server02</name>
    <estado>1</estado>
    <checkable>yes</checkable>
    <url>localhost</url>
    <puerto>5432</puerto>
     <db>ServerDirectorio</db>
    <user>postgres</user>
    <pwd>andreszan</pwd>
    <timeout>3</timeout>
  </sl>
  <sl>
    <name>Server03</name>
    <estado>1</estado>
    <checkable>yes</checkable>
    <url>localhost</url>
    <puerto>5432</puerto>
    <db>ServerRegistro</db>
    <user>postgres</user>
    <pwd>andreszan</pwd>
    <timeout>3</timeout>
  </sl>
  </urls>
```

Imagen 8. **Configuración y conexión a las bases de datos**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qd0pNcTB3SUlIVDA/view?usp=sharing

- Para ejecutar las queries, se deben seleccionar las BDs o servidores a consultar. Ejemplo:

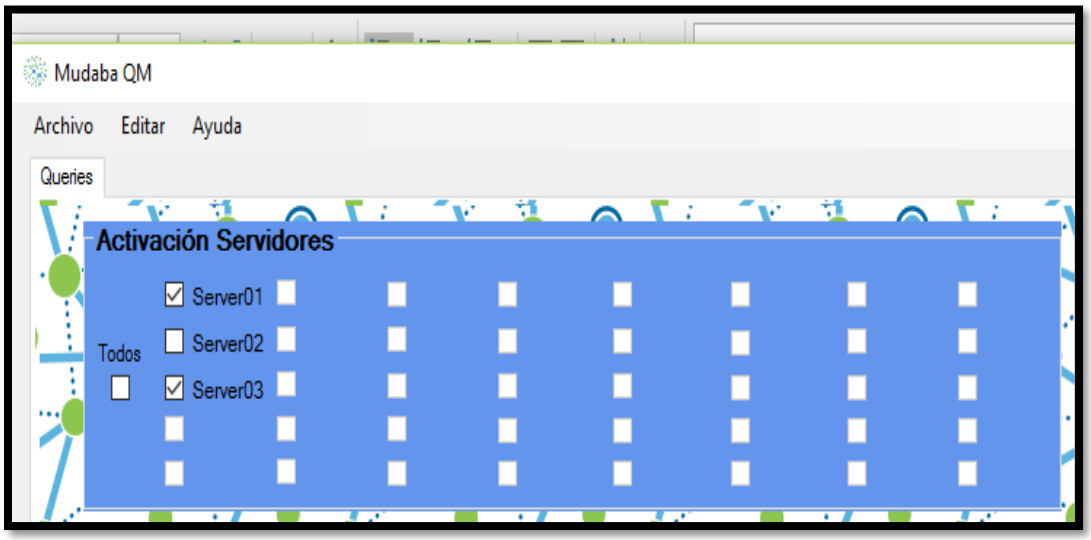


Imagen 9. **Activación servidores (BD)**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qeklQUmR5azZEdHc/view?usp=sharing

- Crear y guardar la colección de queries de uso regular al usar Mudaba QM en el archivo queries.xml. Se guardan con un nombre y su correspondiente query asociada. Ejemplo:

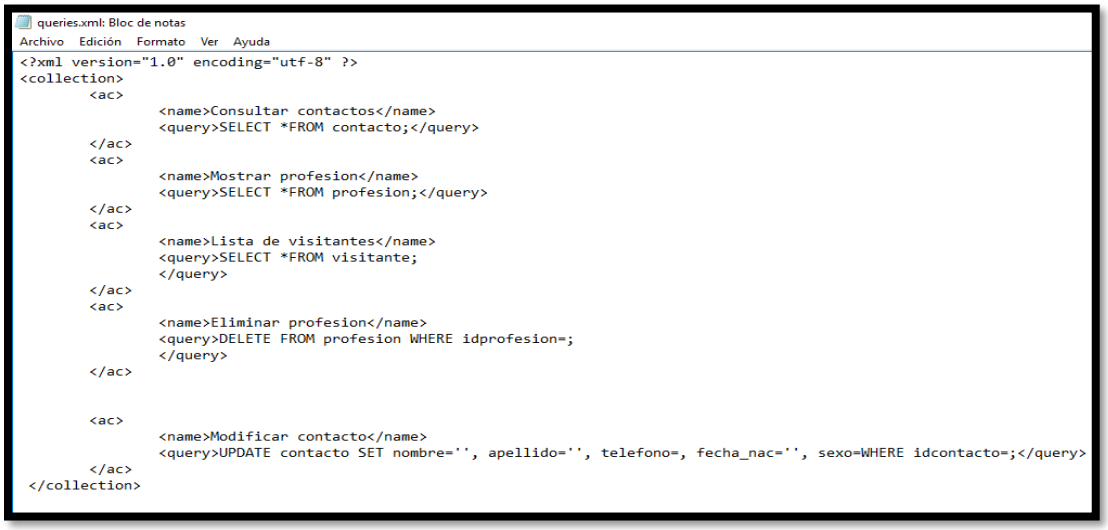


Imagen 10. **Crear y guardar Queries para el autocompletar.**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qalNpUk8xZ3Fzb3M/view?usp=sharing

- Para comprobar que las queries quedaron guardadas, se cierra la ventana actual y se abre otra, se digita en el Texbox las primeras letras de la query según el nombre como se haya guardado para usar la opción autocompletar. Ejemplo:

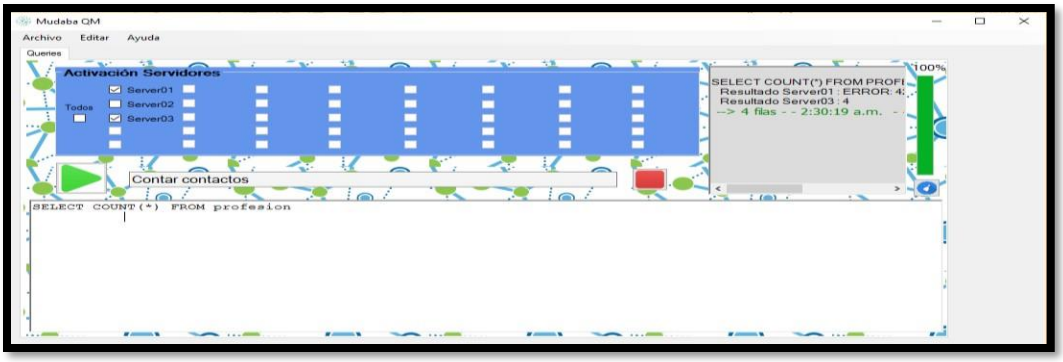


Imagen 11. **Uso de la opción autocompletar**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qblJIVExud3d3SHc/view?usp=sharing

- Automáticamente al hacer la consulta, se muestran los resultados de la misma. Si la consulta es un SELECT, dichos resultados se guardan en un archivo CSV. El directorio contenedor se abrirá automáticamente, o también puede abrirse desde el menú *Archivo* en la opción Carpeta Outputs.

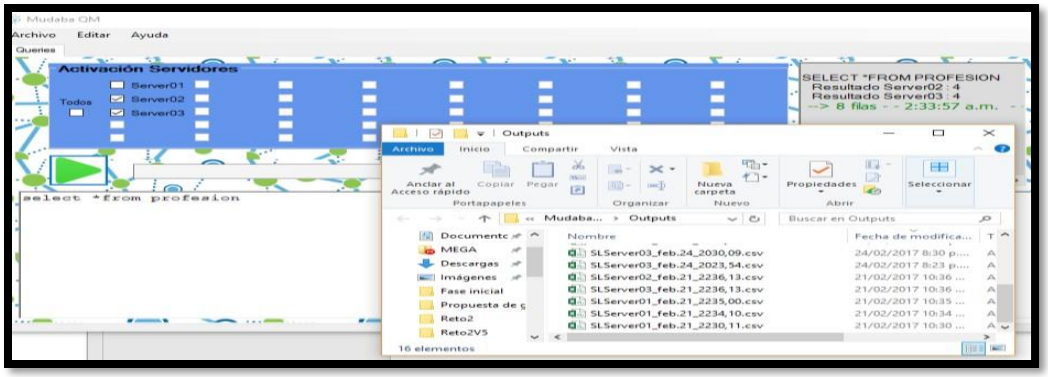


Imagen 12. **Información de la consulta es generada en archivo CSV.**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qUU05VU1JbEdTLVE/view?usp=sharing

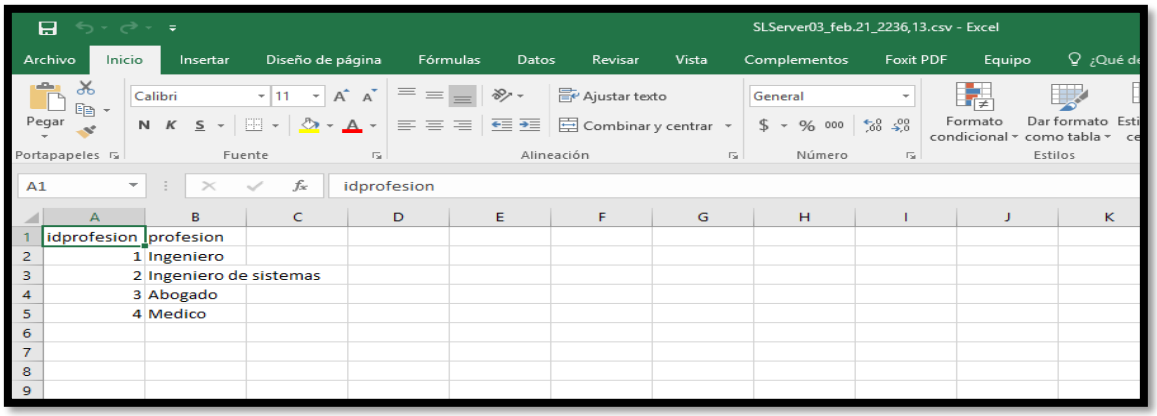


Imagen 13. **Información del archivo CSV mostrada en Excel.**
Fuente: Elaboración propia. Recuperado de: https://drive.google.com/file/d/0B2qtUbgdxp7qbzRaMEJOLVFld1E/view?usp=sharing

9.2. Resultados comparativos de velocidad en las transacciones sobre bases de datos.

Para realizar las comparaciones se tuvo en cuenta los siguientes supuestos:

- Solo se toma en cuenta el motor de bases de datos PostgreSQL
- Se ejecutan consultas para bases de datos iguales en cada servidor.
- Los servidores y bases de datos están previamente creados
- Los permisos de acceso a los diferentes servidores están previamente establecidos en el archivo de configuración de PostgreSQL

TRANSACCIÓN (Prueba hecha sobre 5 servidores simultáneos, el resultado es el promedio de 3 ejecuciones)	**TEMPO DE TRANSACCIÓN** (en segundos)		
	PGADMIN-POSTGRESQL	**MUDABA QM**	**% Mejora**
INSERT (ingresar 100 mil regs.)	157,24	46,17	340,6%
UPDATE (actualizar 100 mil regs.)	29,65	13,4	221,3%
SELECT (consultar y guardar 100 mil regs.)	73,59	3,11	2366,2%
DELETE (borrar 100 mil regs.)	29,1	7,88	369,3%
SELECT COUNT (contar 100 mil regs.)	14,95	1,21	1235,5%

Tabla 4. **Resultados comparativos**
Fuente: Elaboración propia

CONCLUSIONES

- Como resultado de los conocimientos adquiridos durante el programa de Tecnología en Desarrollo de Software, además de las investigaciones independientes relacionadas con el tema POO y Bases de Datos Relacionales, obtenemos como resultado MudabaQM; aplicación que puede sustituir a un software Gestor de Bases de datos en sus principales funciones y que, además, contiene características adicionales como las consultas simultáneas y la autocompletación de *queries*.

- Con la aplicación desarrollada es técnicamente factible ejecutar simultáneamente consultas SQL que se distribuyen a través de la red hacia diferentes servidores interconectados, indicando el resultado de cada ejecución y permitiendo guardar las consultas en un archivo CSV localmente desde el cliente en el que se ejecuta. La aplicación cumple así con su objetivo de ser una alternativa eficiente y multipropósito para gestionar sistemas de bases de datos.

- Los costos del producto desarrollado son viables; la aplicación es de gran utilidad y permite la mejora en la eficiencia al realizar consultas SQL en diferentes servidores. Dicha eficiencia es mucho más notoria cuando los diferentes servidores almacenan bases de datos idénticas en donde se puede evidenciar la mejora en la velocidad de ejecución de las consultas.

- Después de las pruebas y el uso continuado en un ambiente empresarial real, se comprueba la hipótesis y la vez se cumple con el objetivo: es posible mejorar la eficiencia en el manejo de la información de una organización con software de bajo costo para la administración de bases de datos.

REFERENCIAS BIBLIOGRÁFICAS

Booch, G; Maksimchuk, Robert; Object Oriented Analysis and Design with Applications (2007). Tercera Edición.

Córdoba J. F (2006). *Patentes Sobre Software. Desafío a La Propiedad Intelectual en la Sociedad de la Información.* Publicado en Hacia una comprensión humana del derecho. Estudios en homenaje a Roberto Suarez Franco", Temis, pp. 123-144. Recuperado el 30 de noviembre de 2016 de: https://www.unisabana.edu.co/fileadmin/Documentos/Derecho/CEDEPI/Patentabilidad_del_Software.pdf

Danforth, S. Tomlinson, C. Type theories and Object Oriented Programming (1998). System Technology Lab. P. 44

Date, C.J. *Introducción a los Sistemas de Bases de datos (2001)*. Ed. Pearson Education. 7ª edición. Recuperado el 23 de diciembre de 2016 de: http://www.freelibros.org/ingenieria/introduccion-a-los-sistemas-de-bases-de-datos-7ma-edicion-c-j-date.html

Hunt, A. y Thomas, D. *The Pragmatic Programmer* (1999). Ed. Addison Wesley.

Microsoft Net Framework. Tomado el 30 de diciembre de 2016 de: https://www.microsoft.com/net

Richardas, M. *Software Architecture Patterns* (2015). Ed. O'Rally Media. Recuperado el 25 de diciembre de 2016 de http://www.twirpx.com/file/1936162/