Ciencia Digital
www.cienciadigital.org

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Evaluación del Rendimiento de Comunicaciones entre las plataformas Java y .NET utilizando un Cifrado Híbrido

Evaluation of Communications Performance between Java and .NET platforms using a Hybrid Encryption

Vivanco Granda Marco Vinicio [1], Benítez Bravo Víctor Hugo[2], Moreano Sánchez Gabriel Vinicio [1] & Benítez Bravo Álvaro Gabriel

Recibido: 22-03-2019 / Revisado: 27-04-2019 /Aceptado: 26-05-2019/ Publicado: 05-06-2019

Abstract. DOI: https://doi.org/10.32/cienciadigital.v3i1.947

The purpose of this research project was to perform the analysis of the optimal performance of a communication system that uses a hybrid encryption protocol and the sockets mechanism, between the JAVA and .NET platforms, applied to the computer backup system for the Technical Office Loja of the Zonal Coordination 7 of the Civil Registry Identification and Certification.

To determine the performance between both platforms, the analysis of each indicator obtained from the FURPS Model was done, with a sample of 335 files distributed among 200 Light, 100 Medium and 35 Heavy, an error percentage of 0.05% obtaining a percentage variation of the 8.30% effective total yield in favor of C # versus Java. For the implementation of the solution, an application was developed using threads, sockets and hybrid encryption protocol using the agile development methodology SCRUM, later the final product is a communication system as a backup tool for the information deployed in the technological infrastructure of the Provincial Technical Office of Loja.

[1]Escuela Superior Politécnica de Chimborazo, Facultad de Mecánica. Riobamba, Ecuador. mvvivancog@gmail.com, gabriel.moreano@espoch.edu.ec
[2]Universidad Tecnológica Israel, Facultad de Ingeniería, Quito, Ecuador. benitezvh@uisrael.edu.ec, benitezag@uisrael.edu.ec

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Keywords: java, c#, performance, rsa, hybrid encryption, furps

Resumen.
El propósito del presente proyecto de investigación fue realizar el análisis del rendimiento óptimo de un sistema de comunicación que utiliza un protocolo de cifrado hibrido y el mecanismo de sockets, entre las plataformas JAVA y .NET, aplicado al sistema informático de respaldos para la Oficina Técnica Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación.

Para determinar el rendimiento entre ambas plataformas, se realizó el análisis de cada indicador obtenidos del Modelo de FURPS, con una muestra de 335 archivos distribuidos entre 200 Livianos, 100 Medianos y 35 Pesados, un porcentaje de error del 0.05% obteniendo una variación porcentual del rendimiento total efectivo de 8,30% a favor de C# frente a Java. Para la implementación de la solución se desarrolló un aplicativo mediante hilos, sockets y protocolo de cifrado hibrido utilizando la metodología de desarrollo ágil SCRUM, posteriormente el producto final es un sistema de comunicación como herramienta de respaldo de la información desplegada en la infraestructura tecnológica de la Oficina Técnica Provincial de Loja.

Palabras claves: java, c#, rendimiento, rsa, cifrado hibrido, furps

Introducción

Establecer un canal de comunicación por el cual un proceso puede enviar o recibir datos de un equipo a otro, para ello se utiliza *Sockets* como un sistema de comunicación entre procesos de diferentes equipos de una red. Un socket generalmente es soportado por lenguajes de programación de alto nivel como (C#, Java, Phyton, etc) el mismo que emplea un protocolo TCP para realizar la conexión y comunicación entre sus terminales.

De acuerdo a los índices de la comunidad de programación TIOBE actualizado hasta enero del 2018, indica la popularidad de los lenguajes de programación, en donde Java encabeza el primer lugar y en quinto puesto esta C# un lenguaje de programación creado por Microsoft para su plataforma .NET., pero *¿Cuál de las dos ofrece mayor rendimiento en la ejecución de sistemas informáticos distribuidos con seguridad de cifrado híbrido aplicados en la transferencia de archivos?*

El presente trabajo tiene como objetivo el análisis de rendimiento entre los lenguajes Java y C# usando socket y un algoritmo de encriptación hibrida, para posterior desarrollar un sistema de comunicación estable y seguro para la Oficina Técnica de Loja para la transferencia y respaldo de la información basado en la plataforma .NET. Para el desarrollo de la aplicación se utilizará la metodología ágil Scrum, la misma que se basa en obtener resultados prontos ante requerimientos cambiantes.

Seguridad

La seguridad de la información tiene que ver con las medidas preventivas y reactivas de los

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

sistemas tecnológicos que permitan proteger y resguardar la información buscando mantener la confidencialidad, disponibilidad e integridad de la información (Soriano, 2016).
Los mecanismos de seguridad son también llamadas herramientas de seguridad es un proceso que implementa uno o más servicios de seguridad. Estos dan soporte a los servicios de seguridad y ejecutan actividades específicas para la protección frente a ataques o resultados del ataque. Los mecanismos de seguridad son. el cifrado, firma digital, control de acceso, integridad de los datos, intercambio de autenticación, tráfico de relleno, control de encaminamiento, notarización y seguridad perimetral (Soriano, 2016).

Criptografía

La criptografía es la creación de técnicas para el cifrado de datos con la confidencialidad de los mensajes. Si la criptografía es la creación de mecanismos para cifrar datos, el criptoanálisis son los métodos para "romper" estos mecanismos y obtener la información. Una vez que nuestros datos han pasado un proceso criptográfico decimos que la información se encuentra cifrada (Corrales, Cilleruelo, & Cuevas, 2014).

El criptoanálisis consiste en la reconstrucción de un mensaje cifrado en texto simple utilizando métodos matemáticos. Por lo tanto, todos los criptosistemas deben ser resistentes a los métodos de criptoanálisis. Cuando un método de criptoanálisis permite descifrar un mensaje cifrado mediante el uso de un criptosistema, decimos que el algoritmo de cifrado ha sido decodificado (Kioskea, 2014).

Un algoritmo criptográfico no es más que una aplicación de herramientas matemáticas usadas en el proceso de cifrado y descifrado. Un algoritmo criptográfico trabaja en combinación con una clave "*una palabra, número o frase"* para cifrar un documento (texto, imagen, música, video, etc) (Soriano, 2016). Estos se clasifican en: algoritmos de clave simétrica o clave secreta, de clave asimétrica o clave pública y algoritmos hash o de resumen. El proceso de cifrado y descifrado de información mediante el uso de una única clave se conoce como *criptografía de clave simétrica* o *cifrado de secreto compartido* (Soriano, 2016), se ha estado utilizando desde la época de los antiguos egipcios. Esta forma de cifrado utiliza una clave secreta, denominado secreto compartido, utiliza la misma clave para cifrar y descifrar los datos. Consiste en que ambos (emisor y receptor) tienen que conocer la clave común, el problema surge con este método porque se tiene que comunicar la clave secreta de una forma segura al receptor, si en la comunicación una tercera persona intercepta la clave, puede leer el mensaje. Los algoritmos de clave simétrica más utilizados son: **DES** y **AES**.

La criptografía de clave pública o criptografía asimétrica apareció para hacer resolver los problemas de seguridad que plantea la criptografía simétrica. Este método resuelve el problema de transmisión de claves que tiene la criptografía de clave secreta mediante el uso de dos claves (***par de claves***) en vez de una sola, utilizando una de ellas para el cifrado, y la otra para el descifrado (Soriano, 2016). Consiste en que una de las claves es de libre distribución (**clave pública**). Por eso, este método de cifrado también se llama el cifrado de clave pública. La segunda clave es la **clave privada** no es distribuible. Es importante señalar

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

que las claves públicas y privadas están relacionadas, pero es prácticamente imposible deducir la clave privada si se conoce la clave pública. El más común es el algoritmo de clave pública **RSA.**

El sistema hibrido es la combinación de las ventajas del cifrado simétrico y asimétrico, ya que el problema de ambos sistemas criptográficos es que el simétrico es inseguro y el asimétrico es lento. Específicamente, el sistema híbrido utiliza un algoritmo de clave pública con el fin de compartir de forma segura la clave usada en el sistema de cifrado simétrico. El texto en claro se cifra utilizando una clave simétrica, dando lugar al criptograma. El emisor envía el criptograma al receptor junto con la clave simétrica utilizada cifrada con la clave pública del destinatario. Dado que el método de reparto es seguro, la clave simétrica se renueva cada sesión, por eso a veces a esta clave se denomina clave de sesión. Esto significa que, si un atacante es capaz de conocer la clave de sesión, sólo sería capaz de leer el mensaje cifrado con esa clave (Soriano, 2016). En la Figura 1 se muestra el proceso de cifrar mediante criptografía de sistema hibrido.

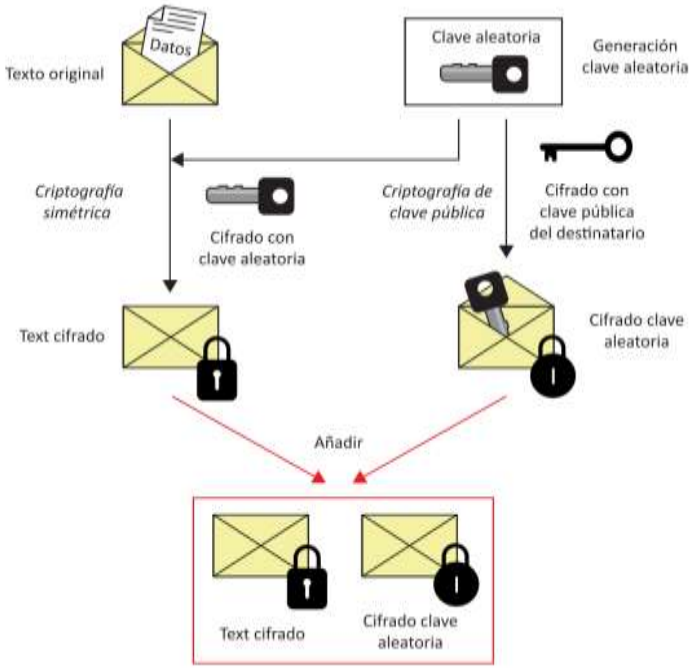


Figura 1. Modelo de Cifrado Hibrido
Fuente: Autores

Arquitecturas cliente-servidor

Una arquitectura cliente/servidor es un modelo de aplicación distribuida en donde las tareas se reparten entre un servidor y un cliente, este modelo es utilizado ampliamente y forma la base en gran medida del uso de las redes. Entre las características relevantes de este modelo están: la interoperabilidad, el rendimiento, la comunicación, ser multiplataforma y la escalabilidad.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Sockets

Los sockets es el canal de comunicación por el cual un proceso puede enviar o receptar información. Fueron popularizados por Berckley Software Distribution, de la universidad norteamericana de Berkley. Usan el protocolo de streams **TCP** (Transfer Contro Protocol) y el de datagramas **UDP** (User Datagram Protocol). Utilizan una serie de primitivas para establecer el punto de comunicación, para conectarse a una máquina remota en un determinado puerto que esté disponible, para escuchar en él, para leer o escribir y publicar información en él, y finalmente para desconectarse (Rojas, Rosas, Roaro, & Puebla, 2012). Existen tres tipos de sockets: Socket Stream, Datagrama y Raw.

El modelo más básico de los sockets mencionado por (Hernanadez, 2006) y (Tangient, 2015) consta de un servidor y un cliente el proceso se detalla a continuación:

1. Un **servidor** se ejecuta sobre una computadora específica y tiene un socket que responde en un puerto específico. El servidor simplemente espera, escuchando a través del socket a que un cliente se conecte con una petición.
2. **En el lado del cliente:** el cliente conoce el nombre de host o la IP del servidor que se encuentra ejecutando y el número de puerto en el cual el servidor está conectado y escuchando. Conociendo esto el cliente realiza una petición de conexión con el servidor.
3. El servidor **acepta la conexión**. Además de aceptar, el servidor obtiene un nuevo socket sobre un puerto diferente. Esto se debe a que necesita un nuevo socket (y, en consecuencia, un numero de puerto diferente) para seguir atendiendo al socket original para peticiones de conexión mientras atiende las necesidades del cliente que se conectó.
4. Por la parte del **cliente**, si la conexión es aceptada, un socket se crea de forma satisfactoria y puede usarlo para comunicarse con el servidor. Es importante darse cuenta de que el socket en el cliente no está utilizando el número de puerto usado para realizar la petición al servidor. En lugar de éste, el cliente asigna un número de puerto local a la máquina en la cual está siendo ejecutado.
5. Ahora el **cliente** y el **servidor** pueden comunicarse escribiendo o leyendo en o desde sus respectivos sockets.
6. Finalmente se cierra o finaliza la conexión.

Plataformas de desarrollo

De acuerdo a los índices de la comunidad de programación TIOBE actualizado hasta noviembre del 2016, indica la popularidad de los lenguajes de programación, en donde Java encabeza el primer lugar y en cuarto puesto esta C# un lenguaje de programación creado por Microsoft para su plataforma .NET, crea aplicaciones web, móviles y de escritorio, por lo que es competencia de Java.

Rendimiento de los sistemas

La corporación **SPEC** (Estándar Performance Evaluation Corporation) propone índices de carácter general, SPECmarks y cuyo significado depende del aspecto en concreto que se esté evaluando. Algunos de estos índices tienen en cuenta *el tiempo de ejecución* de un conjunto de programas y utilizan además algún tipo de normalización y cálculo de medidas para

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

reducir el rendimiento a un único indicador. El tiempo de ejecución de un programa representa la medida exacta del rendimiento de un computador aquel que ejecute la misma cantidad de trabajo en menor tiempo posible será el más rápido (Molero & Ruiz, 2004).
Los modelos calidad resultan de utilidad para la predicción de confiabilidad y en la gerencia de calidad durante el proceso de desarrollo, así como para efectuar la medición del nivel de complejidad de un sistema de software. Estos modelos son: McCall (1977), FURPS (1987), DROMEY (1996) y ISO/IEC 9126 (1991) (Camacho, Cardeso, & Nuñez, 2004).
De todos los modelos el que mejor se adapta al rendimiento de productos de software es el modelo de FURPS específicamente con el factor de calidad de Rendimiento en donde utiliza algunos atributos como: Velocidad de procesamiento, tiempo de respuesta, consumo de recursos, rendimiento efectivo total, eficiencia, atributos esenciales para medir el rendimiento de las plataformas Java y .Net.

Metodología

Tipo y diseño de investigación

Tipo de investigación

La presente investigación puede clasificarse de dos tipos: Aplicativa y cuasi-experimental.

- *Aplicativa:* ya que se basa en conocimientos existentes, derivados de investigaciones previas, dirigida al desarrollo tecnológico para establecer nuevas herramientas para mejorar los procesos existentes de la Oficina Técnica de Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación.
- *Experimental:* ya que utilizará el conocimiento para realizar la integración de tecnologías y el cifrado hibrido para intercambio de datos, bajo las plataformas Java y Microsoft .NET.

Diseño de investigación

El presente tipo de trabajo de investigación es cuasi-experimental. En donde se propone diseñar un sistema de comunicación desarrollado en las plataformas Java y .Net y probar su rendimiento, en el cual se realiza transferencia de archivos de equipos clientes hacia el servidor gestor de respaldo.

Enfoque y alcance investigativo

El presente trabajo investigativo tomará un enfoque cualitativo cuantitativo y un alcance de investigación descriptiva.

Población de estudio

Para determinar la población de estudio se toma como referencia el libro (Pressman, 2002) en donde considera dos tipos de pruebas de rendimiento (*Pruebas de Carga y Pruebas de Esfuerzo*), para la presente investigación se considera de la siguiente manera:

- Cantidad de usuarios o funcionarios de la Oficina Técnica de Loja de la Coordinación

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Zonal 7 de Registro Civil Identificación y Cedulación.

- Número de transacciones (NT) que realiza cada usuario para respaldar la información que puede ser diario, semanal o mensual.

Para determinar el número de transacciones de respaldo de información se considera realizar a diario, es decir cada usuario va a realizar el respaldo de la información dos veces al día, en la Tabla 1 se muestra la población de estudio.

Tabla 1. Población de estudio

CANTÓN	DEPARTAMENTOS USUARIOS		# NT * USUARIO	TOTAL
Loja	Administrativos	17	2	34
	Operativa	26	2	52
Agencias Cantonales	Jefatura/Módulos	12	2	24
Total		**55**		**110**

Fuente: Autores

Selección de la muestra

Para la selección de la muestra se considera utilizar la misma población de estudio, sin embargo, para mejorar la visualización de los resultados del rendimiento se establecerá una de muestra de 335 archivos, el mismo que cumple con el Teorema del Limite Central [*Campana de Gauss*] en la cual establece que para un número de pruebas n mayor de treinta tendrá un comportamiento de distribución normal (Wackerly, Mendenhall, & Scheaffer, 2010); en base a esta muestra se realizarán las pruebas de rendimiento para ambos sistemas tanto en Java como en .NET.

Métodos, técnicas e instrumentos

- **Método científico:** Que consta de varias etapas para obtener un conocimiento válido desde el punto de vista científico, utilizando para esto instrumentos que resulten fiables, consta de las siguientes etapas:
  - Planteamiento del problema que es objeto principal de nuestro estudio.
  - El apoyo del proceso previo a la formulación de la hipótesis.
  - Levantamiento de la información necesaria
  - Análisis e interpretación de Resultados
  - Proceso de la Comprobación de la Hipótesis

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

- Difusión de resultados

- **Método deductivo:** Debido que al estudiar en forma general los diferentes tipos de cifrado de datos, manejo de socket y herramientas de medición de rendimiento, se tratará de encontrar el más adecuado que contenga las mejores características para la seguridad y envió de la información.
- **Método comparativo:** Después del estudio general se deberá comparar cada uno de los tipos de cifrado de datos, manejo de socket y herramientas de medición de rendimiento a estudiarlos.
- **Método analítico-sintético:** que permitió descomponer la variable dependiente rendimiento en sus diferentes indicadores donde se estudió el comportamiento de ambos lenguajes en cada uno, verificando la diferencia significativa y variación porcentual, para luego mediante la síntesis consolidar el resultado en el rendimiento total efectivo.

Técnicas

- *Búsqueda de información*: permite obtener la información necesaria acerca del objeto de estudio de la investigación para su desarrollo, utilizando las fuentes secundarias disponibles.
- *Pruebas:* permite realizar las pruebas para analizar el rendimiento de las 2 aplicaciones del sistema de respaldo.
- *Observación:* permite determinar resultados de las pruebas realizadas en ambas plataformas de desarrollo java y .net.
- *Análisis:* permite determinar los resultados de la investigación.

Instrumentos

Los instrumentos para procesar los datos de los indicadores son los siguientes:

- *Netbeans:* entorno de Desarrollo Integrado (IDE) de código abierto. Permite el desarrollo aplicación en Java con el manejo de socket y el mecanismo de cifrado hibrido de los datos.
- *Visual Studio:* entorno de Desarrollo Integrado (IDE) de código licenciado. Permite el desarrollo aplicación en C# con el manejo de socket y el mecanismo de cifrado hibrido de los datos.
- *Herramientas de rendimiento*: Para el análisis del rendimiento del equipo al momento de realizar la transferencia de archivos.
- Hoja de cálculo Excel.
- Interpretación de los resultados.

Planteamiento de hipótesis

A continuación, se plantea la hipótesis alternativa y la hipótesis nula:

- **Ha=** La plataforma JAVA ofrece mejor rendimiento que la plataforma .NET en el

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

desarrollo de un sistema de comunicación que utiliza un cifrado híbrido, sockets aplicado al Sistema Informático de Respaldos de la Oficina Técnica de Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación.

- **Ho=** La plataforma JAVA NO ofrece mejor rendimiento que la plataforma .NET en el desarrollo de un sistema de comunicación que utiliza un cifrado híbrido, sockets aplicado al Sistema Informático de Respaldos de la Oficina Técnica de Loja de la Coordinación Zonal 7 de Registro Civil Identificación y Cedulación.

Variables e indicadores

En base a la hipótesis de investigación planteada en el presente trabajo la cual dicta: *La plataforma JAVA ofrece mejor rendimiento que la plataforma .NET en el desarrollo de un sistema de comunicación que utiliza un cifrado híbrido*, se determinan las siguientes variables:

- **Variable Independiente**
  - La plataforma utilizada (Java y .Net)

- **Variable Dependiente**
  - El rendimiento que ofrece las plataformas Java y .Net para el sistema de comunicación que utiliza un cifrado híbrido, sockets aplicado al Sistema Informático de Respaldos.

Operacionalización de variables

Teniendo la variable dependiente e independiente se realizó la operacionalización de ambas variables para desarrollar un aplicativo gestor de respaldo en las plataformas Java o .Net y evaluar el rendimiento de ambas.

Tabla 2. Operacionalización variable independiente

VARIABLES	INDICADORES	INDICES
Independiente	• Lenguajes	• Lenguaje utilizado
Plataforma utilizada (Java y .Net)	• Tipo de archivos	• Livianos • Medianos • Pesados

Fuente: Autores

Los indicadores de la variable dependiente (VD) planteados permitieron determinar el rendimiento entre los prototipos Java y C#, estos indicadores fueron tomados del Modelo de FURPS de Pressman (Pressman, 2002) el mismo que hace mención de la evaluación del rendimiento de software.

Procesamiento y análisis

La información relacionada a la investigación es analizada y presentada en figuras o gráficos

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

estadísticos utilizando el sistema SIAE 2.0 (Salazar & Alvarez, 2017) y aplicando la estadística inferencial con la prueba Z con una muestra de 335 archivos, para determinar la comprobación o negación de la hipótesis.

Tabla 3. Operacionalización variable dependiente

VARIABLES	INDICADORES	INDICES
Dependiente El rendimiento que ofrece las plataformas Java y .Net para el sistema de comunicación que utiliza un cifrado híbrido, sockets aplicado al Sistema Informático de Respaldos.	• Tiempos de Respuestas	• Cantidad de segundos por tarea.
	• Velocidad de Procesamiento	• Numero de archivos por milisegundo
	• Consumo de Recursos	• Porcentaje de uso de CPU • Porcentaje de uso de RAM
	• Eficiencia	• Número de transferencia de archivos cumplidas satisfactoriamente.

Fuente: Autores

Parámetros y herramientas

Para al análisis del rendimiento se realizó las pruebas de envió de archivos en los prototipos Java y C# con una muestra de 335 archivos y considerando la cantidad, tipos y tamaños de archivos que se ocupan en los procesos diarios de la Coordinación de Oficina Técnica de Loja se realizó un análisis y determinación del porcentaje de utilización de los archivos.

Tabla 4. Ponderación de la muestra

Archivos Livianos (50%) (1MB –2MB)	Archivos Medianos (30%) (100MB – 200MB	Archivos Pesados (20%) (1GB – 2GB)	Total
200	100	35	335

Fuente: Autores

Método FURPS

El método FURPS (Constanzo, 2014) dentro de sus factores de calidad se encuentra el Rendimiento cuyos atributos como: Velocidad de Procesamiento (VR), Tiempo de Respuesta (TR), Consumo de Recursos (CR), Eficiencia (E) y Rendimiento Efectivo Total (RET), atributos esenciales para medir el rendimiento de las plataformas Java y .Net.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Se definió el porcentaje de ponderación de 25% para cado atributo VR, TR, CR y E porque se consideró que cada atributo tiene el mismo nivel de importancia para obtener el rendimiento de cada prototipo.

Tabla 5. Ponderación de indicadores FURPS

Atributos	Porcentaje de ponderación
Velocidad de Procesamiento (VR)	25%
Tiempo de Respuesta (TR)	25%
Consumo de Recursos (CR)	25%
Eficiencia (E)	25%
TOTAL	**100%**

Fuente: Autores

Finalmente, para obtener el rendimiento se utiliza la estadística descriptiva para los prototipos Java y C# y se aplicó esta fórmula.

Fórmula 1: Rendimiento para Java y C#

$$\boldsymbol{VPRT} = \text{VPJ} - \text{VPC\#} \quad \text{o} \quad \text{VPC\#} - \text{VPJ}$$

Donde:

- VPRT = Variación Porcentual del Rendimiento Total
- VPJ = Variación Porcentual de Java
- VPC# = Variación Porcentual de C#

Ahora se debe determinar la fórmula para sacar VPJ y VPC#

$$\boldsymbol{VPJ\ o\ VPC\#} = VPTR + VPVP + VPCR + VPE$$

Donde:

- VPTR = Variación Porcentual del Tiempo de Respuesta
- VPVP = Variación Porcentual de la Velocidad de Procesamiento
- VPCR = Variación Porcentual del Consumo de Recursos
- VPE= Variación Porcentual de la Eficiencia

Herramientas

Las herramientas a utilizar para demostrar el rendimiento entre las plataformas Java y .Net en una arquitectura cliente-servidor utilizando un algoritmo de cifrado hibrido y sockets se clasificó así:

Para evaluación de prototipos

Las herramientas el desarrollo de los dos prototipos tanto para Java como C# son:

- *Netbeans 8.2:* entorno de Desarrollo Integrado (IDE) de código abierto.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

- *Visual Studio 2015:* entorno de Desarrollo Integrado (IDE) de código licenciado. La versión de Visual Studio 2015 Enterprise se utilizó.

Para evaluación del rendimiento

Las herramientas para demostrar el rendimiento en la transferencia de archivos utilizando los prototipos para Java como C# son:

- *VisualVM:* Se utilizo esta herramienta para determinar el consumo de recursos (*Memoria RAM y CPU*) para Java.
- *Herramienta de Diagnostico Visual Studio 2015:* Se utilizo la misma herramienta de diagnóstico de Visual Studio 2015 que muestra en tiempo de ejecución el consumo de recursos (Memoria RAM y CPU) para C#.
- *Excel 2016:* Se utilizo el paquete de office 2016, específicamente Microsoft Excel 2016 para obtener las estadísticas entre los tiempos de Java y C#.

Para tabulación y análisis estadísticos

- Herramienta para calcular la hipótesis: Se utilizó el Sistema Inteligente de Análisis Estadístico (SIAE) para comprobar la hipótesis y aplicando la estadística inferencial con la prueba Z.

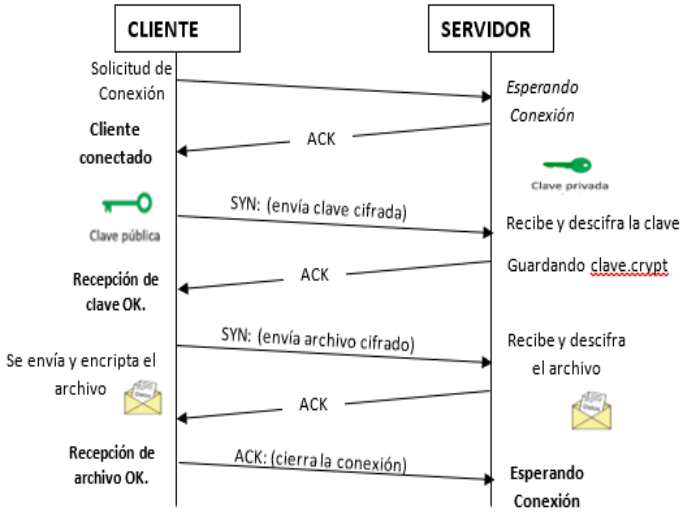
Prototipos

Para demostrar cual plataforma Java y .Net es más eficiente en arquitecturas cliente-servidor utilizando un algoritmo de cifrado hibrido y sockets se desarrolló dos escenarios tanto en Java y C# realizando pruebas de envió de 200 archivos de tamaño liviano, 100 medianos y 35 pesados.

Escenario propuesto

Para el desarrollo y demostración de la solución se propone el siguiente escenario aplicado para los prototipos Java y C# para una arquitectura cliente-servidor.

Figura 2. Escenario arquitectura cliente-servidor para Java y C#



Fuente: Autores

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Hardware cliente/servidor

En la Tabla 6se muestra el hardware utilizado tanto para el Cliente como el Servidor en el que se instaló y configuro los prototipos desarrollados para las pruebas de rendimiento entre Java y C#.

Tabla 6. Características de Hardware

Características	Cliente	Servidor
Sistema Operativo	Win10	Win7
Marca	Dell	Clon
Procesador	Intel i7	Intel Core 2
RAM	8GB	2GB
Disco Duro	2TB	500GB
Internet	10MB	10MB

Fuente: Autores

Software utilizado

En la Tabla 7 se muestra el software utilizado para el desarrollo de los 2 prototipos Java y C#.

Tabla 7. Características de Software

Características	Java	C#
Plataforma	Netbeans 8.2	Visual Studio 2015
Framework	Java Se 8	.NET Framework 4.6.1
Librerías Criptográficas	JCE	Bouncy Castle
Librerías Sockets	Java.Net	System.Net

Fuente: Autores

Resultados

En esta sección se realiza la recolección y análisis de datos por cada indicador obteniendo la variación porcentual de la plataforma que ofrece mejores resultados.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Indicador: Velocidad de procesamiento

Para el desarrollo de este indicar se lo hizo en base a los resultados del promedio del tiempo de envió en milisegundos, moda, mediana y desviación típica de los 335 archivos utilizado como muestra y distribuidos para livianos (200), medianos (100) y pesados (35) en los prototipos Java como C#.

Tabla 8. Porcentaje de velocidad de procesamiento para Java y C#

INDICADOR	LENGUAJE DE PROGRMACIÓN		DS	MEJOR LENG.	VARIACIÓN PORCENTUAL	
	JAVA	C#			VALOR ABS.	VALOR REL.
	VP (ms)	VP (ms)				
LIVIANOS (50%)	594	225	SI	C#	62,12	31,06%
MEDIANOS (30%)	38347	25749	SI	C#	32,85	9,86%
PESADOS (20%)	630539	323234	SI	C#	48.74	9,75%
VALOR TOTAL						**50,67%**

Fuente: Autores

Luego de obtener los resultados para demostrar sí existe una diferencia significativa (DS) tanto para archivos livianos, medianos y pesados entre Java y C# se utilizó el software SIAE (Salazar & Alvarez, 2017) para el análisis de la hipótesis empleando el método de estadística inferencial con la prueba Z y un margen de error del 5%.

En la Tabla 8 se muestran los resultados de la variación porcentual tanto para Java como C# para los resultados de Archivos livianos, medianos y pesados.

Se concluye que existe un **50,67%** de variación porcentual a favor del aplicativo **C#** frente al aplicativo Java en cuanto a la velocidad de procesamiento.

Indicador: Tiempo de respuesta

Para el desarrollo de este indicar se lo hizo en base a los resultados de los tiempos de respuesta en segundos, moda, mediana y desviación típica de los 335 archivos utilizado como muestra y distribuidos para livianos (200), medianos (100) y pesados (35) en los prototipos Java como C#.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Luego de obtener los resultados para demostrar sí existe una diferencia significativa (DS) tanto para archivos livianos, medianos y pesados entre Java y C# se utilizó el software SIAE [14] para el análisis de la hipótesis empleando el método de estadística inferencial con la prueba Z y un margen de error del 5%. En la Tabla 9 se muestran los resultados de la variación porcentual tanto para Java como C# para los resultados de Archivos livianos, medianos y pesados.

Tabla 9. Porcentaje de tiempo de respuesta para Java y C#

INDICADOR	LENGUAJE DE PROGRMACIÓN		DS	MEJOR LENG.	VARIACIÓN PORCENTUAL	
	JAVA	C#			VALOR ABS.	VALOR REL.
	VP (s)	VP (s)				
LIVIANOS (50%)	88	47	SI	C#	87,23	43,62%
MEDIANOS (30%)	3521	2647	SI	C#	33,02	9,91%
PESADOS (20%)	16364	15168	SI	C#	7,89	1,58%
VALOR TOTAL						**55,11%**

Fuente: Autores

Se concluye que existe un **55,11%** de variación porcentual a favor del aplicativo **C#** frente al aplicativo Java en cuanto al tiempo de respuesta.

Indicador: Consumos de recursos

Para el desarrollo de este indicar se lo hizo en base a los resultados de la cantidad de memoria RAM y CPU utilizada, moda, mediana y desviación típica de los 335 archivos utilizado como muestra y distribuidos para livianos (200), medianos (100) y pesados (35) en los prototipos Java como C#.

Luego de obtener los resultados para demostrar sí existe una diferencia significativa (DS) tanto para archivos livianos, medianos y pesados entre Java y C# se utilizó el software SIAE [14] para el análisis de la hipótesis empleando el método de estadística inferencial con la prueba Z y un margen de error del 5%.

En la Tabla 10 y 11 se muestran los resultados de la variación porcentual de la memoria RAM y CPU utilizado tanto para Java como C# para los resultados de Archivos livianos, medianos y pesados.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

Tabla 10. Porcentaje de memoria RAM utilizada para Java y C#

INDICADOR	LENGUAJE DE PROGRMACIÓN		DS	MEJOR LENG.	VARIACIÓN PORCENTUAL	
	JAVA	C#			VALOR ABS.	VALOR REL.
	Memoria RAM (MB)	Memoria RAM (MB)				
LIVIANOS (50%)	16	21	SI	JAVA	31,25	15,63%
MEDIANOS (30%)	17	19	SI	JAVA	11,77	3,53%
PESADOS (20%)	20	18	NO	NINGUNO	0	0%
VALOR TOTAL						**19,16%**

Fuente: Autores

Tabla 11. Porcentaje de CPU utilizada para Java y C#

INDICADOR	LENGUAJE DE PROGRMACIÓN		DS	MEJOR LENG.	VARIACIÓN PORCENTUAL	
	JAVA	C#			VALOR ABS.	VALOR REL.
	CPU (%)	CPU (%)				
LIVIANOS (50%)	1	6	SI	JAVA	83,33	41,67%
MEDIANOS (30%)	6	9	SI	JAVA	33,33	10,00%
PESADOS (20%)	5	7	NO	NINGUNO	0	0%
VALOR TOTAL						**51,67%**

Fuente: Autores

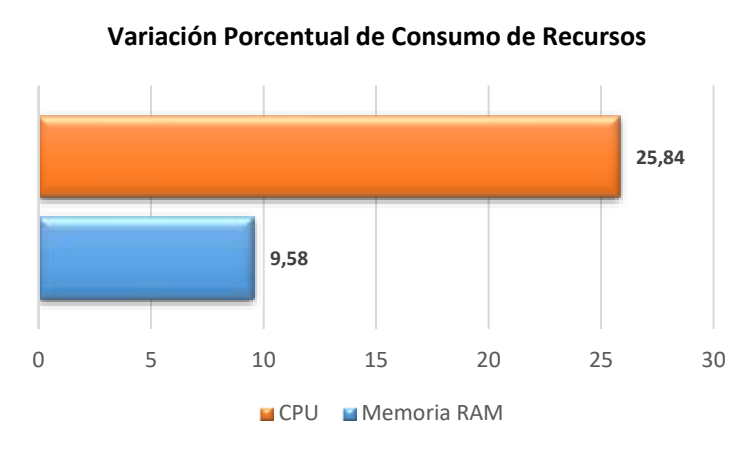
ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

De acuerdo con los resultados obtenidos en las Tablas 10 y 11 en donde se observan la variación porcentual de la cantidad de memoria RAM utilizada y el porcentaje de CPU utilizado entre Java y C# para la transferencia de archivos livianos, medianos y pesados, por lo tanto, se procede a la valoración en porcentaje de ambas variaciones porcentuales cuyos resultados se observan en la Tabla 12 y en la Figura 3.

Tabla 12. Variación de porcentaje en consumo de recursos de Java y C#
Fuente: Autores

INDICADOR	LENGUAJE GANADOR	VARIACIÓN PORCENTUAL	
		VALOR ABSOLUTO	VALOR RELATIVO
MEMORIA RAM (50%)	JAVA	19,16	9,58%
CPU (50%)	JAVA	51,67	25,84%
VALOR TOTAL			**35,42%**

Figura 3. Variación de porcentaje en consumo de recursos de Java y C#



Fuente: Autores

Se concluye que existe un **35,42%** de variación porcentual a favor de **Java** frente a C# en cuanto al consumo de recursos.

Indicador: Eficiencia

Para el desarrollo de este indicar se lo hizo en base a los resultados del promedio de transacciones satisfactorias del proceso de envió/recepción de 335 archivos distribuidos para livianos (200), medianos (100) y pesados (35) utilizando los prototipos Java y C# obteniendo la Tabla 13.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

En el proceso de envió de 335 archivos distribuidos para livianos (200), medianos (100) y pesados (35) del cliente al servidor mediante red se pudo observar que cada una de las transacciones se cumplieron satisfactoriamente tanto en Java como en C#.
Se concluye que existe un **0%** de variación porcentual entre la eficiencia de **Java** y **C#**.

Tabla 13. Resultados de promedio de transacciones satisfactorias de Java y C#

INDICADOR	LENGUAJE DE PROGRMACIÓN		DS	MEJOR LENG.	VARIACIÓN PORCENTUAL	
	JAVA	C#			VALOR ABS.	VALOR REL.
Promedio de transacciones realizadas satisfactoriamente (100%)	335	335	NO	NINGUNO	0	0%
VALOR TOTAL						**0%**

Fuente: Autores

Rendimiento total efectivo

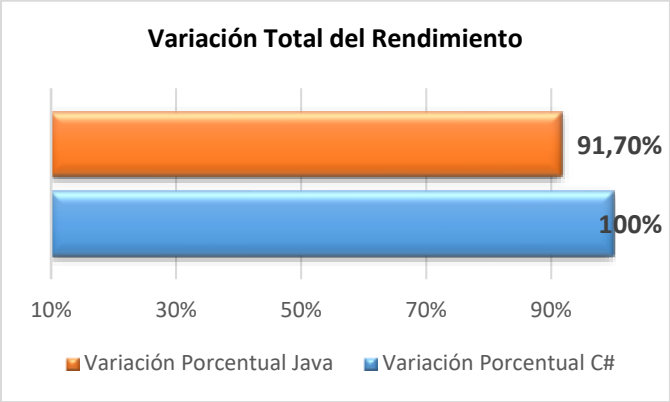
Se procede con el cálculo del rendimiento total efectivo que consiste en obtener el porcentaje relativo y absoluto para Java y C# obteniendo la Tabla 14 y visualizados en la Figura 4.

Tabla 14. Rendimiento total efectivo de Java y C#

INDICADOR	MEJOR LENG.	VARIACIÓN PORCENTUAL			
		VALOR ABS. JAVA	VALOR REL. JAVA	VALOR ABS. C#	VALOR REL. C#
Tiempo de respuesta (25%)	C#	0	0	55,11	**13,78%**
Velocidad de procesamiento (25%)	C#	0	0	13.5	**3,38%**
Consumo de recursos (25%)	JAVA	35,42	**8,86%**	0	0
Eficiencia (25%)	**NINGUNO**	0	**0**	0	0
Rendimiento Efectivo Total			**8,86%**		**17,16%**

Fuente: Autores
Figura 4. Rendimiento total efectivo de Java y C#

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019



Fuente: Autores

Se concluye que existe un **8,30%** de variación porcentual del rendimiento total efectivo a favor del aplicativo **C#** frente al aplicativo Java.

Comprobación de la hipótesis

Aplicando el método de estadística descriptiva se concluye que el aplicativo **C#** ofrece un mejor rendimiento con **100%** frente al aplicativo Java con un 91,70%, existiendo una diferencia de **8,30%,** por lo cual se rechaza la hipótesis alternativa y se acepta la hipótesis nula.

Conclusiones

- El realizar las pruebas de transferencias de archivos 200 livianos, 100 medianos y 35 pesados permitió conocer los tiempos en milisegundos que conlleva cada proceso tanto para Java y C# ejecutándose las mismas sin inconvenientes.
- Una vez realizada las mediciones de cada indicador se obtuvieron los siguientes resultados: para el tiempo de respuesta existe un 55,11% de variación porcentual a favor del aplicativo C# frente al aplicativo Java, para la velocidad de procesamiento existe un 50,67% de variación porcentual a favor del aplicativo C# frente al aplicativo Java, así mismo, para el consumo de recursos existe un 35,42% de variación porcentual a favor de Java frente a C#, en cuanto a la eficiencia, no se encontró una variación porcentual debido a que todas las transacciones se ejecutaron eficientemente, llegando a la conclusión de que en tres los indicadores de rendimiento, la plataforma C# obtuvo un mejor o igual resultado que Java.
- Mediante el análisis de los indicadores se pudo observar que el lenguaje C# ofrece un mejor rendimiento que Java; por lo tanto, se concluye que existe un 8,30% de variación porcentual total a favor del aplicativo C# frente a Java.
- La implementación del sistema de respaldos desarrollado para la Oficina Técnica de Loja del Registro Civil mediante la plataforma C# incorpora una arquitectura cliente/servidor aplicando un algoritmo de cifrado hibrido y usando socket e hilos como también una base de datos ofrece una buena alternativa para el respaldo de la información segura.

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

- **Trabajos Futuros**

    - Se recomienda migrar a entorno web el sistema de respaldos e sincronizado con el active directory y la base de datos de la institución, además investigar sobre los mecanismos de seguridad en autenticación y firmado mediante certificados digitales que poseen cada uno de los frameworks Java y .Net.

Referencias bibliográficas.

Camacho, E., Cardeso, F., & Nuñez, G. (2004). *Arquitecturas de Software: Guias de Estudio*.

Constanzo, M. (2014). *COMPARACION DE MODELOS DE CALIDAD, FACTORES Y METRICAS EN EL AMBITO DE LA INGENERIA DE SOFTWARE*. Retrieved from https://ebookcentral.proquest.com/lib/bibliocauladechsp/reader.action?docID=3206903&query=ingenieria+de+software#

Corrales, H., Cilleruelo, C., & Cuevas, A. (2014). *Criptografía y Métodos de Cifrado*.

Hernanadez, C. (2006). *Herramientas y Lenguajes de Programación*.

Kioskea. (2014). *Criptografía*.

Molero, X., & Ruiz, C. (2004). *Evaluación y Modelado del Rendimiento de los Sistemas Informáticos*.

Pressman, R. (2002). *Ingeniería de Sofware: Un Enfoque Práctico (Quita Edición ed.*

Rojas, M., Rosas, V., Roaro, R., & Puebla, J. (2012). *Programación Paralela y Concurrente.*

Salazar, N., & Alvarez, A. (2017). SIstema Integral de Análisis Estadístico 2.0. *Academia*.

Soriano, M. (2016). *Seguridad en redes y seguridad en la información.*

Tangient, L. L. C. (2015). Programación de redes de Telecomunicaciones. *Obtenido de HttpsprogramacionderedesdetelecomunicacioneswikispacescomProgramaciondeSockets*.

Wackerly, D., Mendenhall, W., & Scheaffer, R. (2010). *Estadística Matemática con Aplicaciones* (7th ed.). Mexico: CENGAGE Learning. Retrieved from https://www.cimat.mx/ciencia_para_jovenes/bachillerato/libros/[Wackerly,Mendenhall,Scheaffer]Estadistica_Matematica_con_Aplicaciones.pdf

ISSN: 2602-8085
Vol. 3, N°2,6 p. 141-161, abril - junio, 2019

PARA CITAR EL ARTÍCULO INDEXADO.

Marco Vinicio, V., Benítez Bravo, V. H., Moreano Sánchez, G., & Benítez Bravo, Álvaro. (2019). Evaluación del rendimiento de comunicaciones entre las plataformas Java y .NET utilizando un cifrado híbrido. *Ciencia Digital*, *3*(2.6), 141-161. https://doi.org/10.33262/cienciadigital.v3i2.6.524