INSTITUTO TECNOLOGICO DE CIUDAD MADERO

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

"POR MI PATRIA Y POR MI BIEN"

TESIS

ANÁLISIS DE ESTRATEGIAS DE INTERCAMBIO DE DATOS PARA COMUNICACIÓN ENTRE APLICACIONES DE OPTIMIZACIÓN

Que para obtener el Grado de

Maestro en Ciencias de la Computación

Presenta

Ing. Nelson Hernández Morales

G16070808

No. CVU: 1170097

Director de Tesis

Dr. Nelson Rangel Valdez

No. CVU: 48135

Co-director de Tesis

Dra. Claudia Guadalupe Gómez Santillán

Cd. Madero, Tamaulipas Marzo 2024

Ciudad Madero, Tamaulipas, 14/diciembre/2023

Oficio No.: U.153/2023
Asunto: Autorización de impresión de tesis

C. NELSON HERNÁNDEZ MORALES
No. DE CONTROL G16070808
P R E S E N T E

Me es grato comunicarle que después de la revisión realizada por el Jurado designado para su Examen de Grado de Maestría en Ciencias de la Computación, se acordó autorizar la impresión de su tesis titulada:

"ANÁLISIS DE ESTRATEGIAS DE INTERCAMBIO DE DATOS PARA COMUNICACIÓN ENTRE APLICACIONES DE OPTIMIZACIÓN"

El Jurado está integrado por los siguientes catedráticos:

PRESIDENTA:	DRA.	LAURA CRUZ REYES
SECRETARIA:	DRA.	MARÍA LUCILA MORALES RODRÍGUEZ
VOCAL:	DR.	NELSON RANGEL VALDEZ
SUPLENTE:	DRA.	CLAUDIA GUADALUPE GÓMEZ SANTILLÁN
DIRECTOR DE TESIS:	DR.	NELSON RANGEL VALDEZ
CO-DIRECTORA:	DRA	CLAUDIA GUADALUPE GÓMEZ SANTILLÁN

Es muy satisfactorio para la División de Estudios de Posgrado e Investigación compartir con usted el logro de esta meta. Espero que continúe con éxito su desarrollo profesional y dedique su experiencia e inteligencia en beneficio de México.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Por mi patria y por mi bien" ®

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE CIUDAD MADERO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

MARCO ANTONIO CORONEL GARCÍA
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
MACG 'MLMR'

Agradecimientos

En el momento de culminar este importante capítulo de mi formación académica, deseo expresar mi sincero agradecimiento a todas las personas e instituciones que contribuyeron de manera significativa a la realización de esta tesis.

En primer lugar, quiero expresar mi profundo reconocimiento al Instituto Tecnológico de Ciudad Madero por brindarme la oportunidad de cursar esta maestría. La educación de calidad y el entorno de aprendizaje en esta institución han sido fundamentales en mi crecimiento profesional tanto en la licenciatura como en esta ocasión en la maestría.

Mi más sincero agradecimiento al Dr. Nelson Rangel Valdez, mi director de tesis, quien no solo me guio en esta etapa, sino que también ha sido un apoyo constante desde mi licenciatura. Sus consejos, orientación y conocimiento han sido invaluables.

Agradezco de igual manera a la Dra. Claudia Guadalupe Gómez Santillán, mi codirectora, por su incansable apoyo, paciencia y disponibilidad a lo largo de este proceso. Su experiencia y dedicación fueron esenciales para la realización de esta tesis.

Así como a mi tutora la Dra. María Lucila Morales Rodríguez que durante estos dos años me guio en todos los pasos a seguir en esta etapa con su paciencia, disponibilidad, consejos y orientación.

A mi comité evaluador, les agradezco por sus valiosos comentarios y sugerencias, los cuales contribuyeron de manera significativa a la mejora de este trabajo. Sus perspicaces observaciones fueron un aporte fundamental en mi proceso de investigación.

No puedo pasar por alto expresar mi profundo agradecimiento al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por su generoso apoyo financiero a través de una beca. Sin su respaldo, esta oportunidad de formación académica no habría sido posible.

Agradezco a mi padre, quien han sido mi fuente constante de motivación y apoyo incondicional en cada etapa de mi vida. Sin su aliento y respaldo, este logro no sería posible.

En resumen, este logro académico es el resultado del esfuerzo colectivo de muchas personas y entidades. A todos ustedes, les estoy profundamente agradecido por su apoyo y confianza en mí.

Gracias.

Resumen

En la vida cotidiana es recurrente que dos aplicaciones de software local, es decir, aplicaciones que no están conectadas a la red, necesiten comunicarse porque cada una de ellas tiene una finalidad diferente y se complementan entre sí. Para conseguir esta comunicación, existen otras aplicaciones conocidas como interfaces de comunicación que permiten la integración de diferentes programas de forma sencilla. Estas interfaces de comunicación están formadas por los formatos de intercambio de datos, algunos de los más conocidos son JSON, XML, CSV, entre otros.

Los formatos de intercambio de datos son herramientas que permiten estructurar la información que se va a transmitir. Algunos de ellos pueden trabajar con ciertos tipos de datos más fácilmente, por ejemplo, el potencial de CSV se encuentra en el manejo de datos en formato de tablas, para XML su potencial está en el manejo de documentos de texto, hojas de cálculo, páginas web y bases de datos, entre otros, y JSON es una alternativa más rápida que XML para datos estructurados.

A la hora del desarrollo, es más fácil utilizar un formato como JSON, CSV, o XML porque ya existen convertidores o generadores de estas estructuras mientras que, utilizando texto plano, estos deben ser desarrollados dependiendo de la estructura de este. Como se puede observar, los formatos de intercambio de datos proporcionan ventajas a los usuarios al permitirles comunicar sus aplicaciones que les permitirán disponer de una ayuda para la toma de decisiones. Es bien sabido que los problemas del mundo real a menudo persiguen más de un objetivo, los cuales tienen diferentes grados de importancia o peso para el o los responsables de la toma de decisiones.

Existen diversas herramientas de software que incluyen métodos que ayudan en el proceso de toma de decisiones; sin embargo, puede resultar complicado para el decisor familiarizarse con nuevos entornos, herramientas o marcos de trabajo para modelar sus problemas. Es por ello que en este trabajo proponemos realizar un análisis del uso de los formatos de intercambio de datos con el objetivo de diseñar una interfaz de comunicación entre un Chat-Bot en telegram, una herramienta de visualización y diagnóstico (VisTHAA) y un optimizador (MS-DOSS), de manera que un tomador de decisiones pueda tener una comunicación directa a través de la integración de herramientas de optimización.

Summary

In everyday life it is common that two local software applications, in other words applications that are not connected to the network, need to communicate because each of them has a different purpose and they complement each other. To achieve this communication, there are other applications known as communication interfaces that allow the integration of different programs in an easy way. These communication interfaces are formed by data interchange formats, some of the best known are JSON, XML, CSV, among others.

Data exchange formats are tools that allow structuring the information to be transmitted. Some of them can work with certain types of data more easily, for example CSV's potential lies in the handling of data tables, for XML its potential is in the handling of text documents, spreadsheets, web pages and databases, among others, and JSON is a faster alternative to XML for structured data.

At the time of development, it is easier to use a format such as XML, JSON, or CSV because there are already parsers or generators of these structures while using plain text, these must be developed depending on the structure of this. As it can be observed, data interchange formats provide advantages to users by allowing them to communicate their applications that will allow them to have an aid for decision making. It is well known that real world problems often pursue more than one objective, which have different degrees of importance or weight for the decision maker(s).

There are several software tools that include methods that help in the decision-making process; however, it can be complicated for the decision maker to become familiar with new environments, tools, or frameworks to model their problems. That is why in this work we propose to make an analysis of data exchange strategies with the objective of designing a communication interface between a Chat-Bot in telegram, a visualization and diagnosis tool (VisTHAA) and an optimizer (MS-DOSS), so that a decision maker can have direct communication through the integration of optimization tools.

Declaración de originalidad

Declaro y prometo que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patentes y similares. Por lo tanto, la obra es de mi autoría y soy titular de los derechos que surgen de la misma.

Declaro también que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y publicaciones.

Además, en caso de presentarse cualquier reclamación o acción por parte de un tercero en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de ésta a mi director y codirectores de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Ciudad Madero y a sus respectivas autoridades.

Nelson Hernández Morales.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la vida cotidiana es habitual que dos aplicaciones de software locales, es decir, aplicaciones que no están conectadas a la red, necesiten comunicarse porque cada una de ellas tiene una finalidad diferente y se complementan. Para conseguir esta comunicación, existen otras aplicaciones conocidas como interfaces de comunicación que permiten la integración de diferentes programas de forma sencilla. Estas interfaces de comunicación están formadas por los formatos de intercambio de datos, algunos de los formatos de intercambio de datos más conocidos son JSON, XML, CSV, entre otros.

Existen diversas herramientas de software que incluyen métodos que ayudan en el proceso de toma de decisiones; En el caso de este proyecto, se cuenta con una herramienta interactiva para apoyar la toma de decisiones en procesos de optimización en presencia de imprecisión e incertidumbre. Este módulo consiste en un módulo de optimización y una interfaz en Telegram donde el usuario establece sus preferencias a través de un proceso interactivo. Inicialmente, se elige un conjunto de soluciones aleatorias para el problema DTLZ, y se selecciona la solución de compromiso (la que más le ha gustado al decisor). A continuación, se establecen las preferencias del decisor mediante un ciclo en el que elige qué soluciones le gustan y cuáles no.

Para comunicar los módulos de la herramienta se utiliza un formato de intercambio de datos, que son herramientas que permiten estructurar la información que se va a transmitir. Algunos de ellos pueden trabajar con ciertos tipos de datos más fácilmente, por ejemplo, el potencial de CSV está en el manejo de tablas de datos, para XML su potencial está en el manejo de documentos de texto, hojas de cálculo, páginas web y bases de datos, entre otros, y JSON es una alternativa más rápida a XML para datos estructurados.

Entre los formatos mencionados, JSON y XML son los más utilizados para la comunicación entre aplicaciones en la web. Esto se debe a que los desarrolladores optan por los formatos ya implementados en el lenguaje que utilizan a la hora de crear estas aplicaciones, ya que les resulta más sencillo; esto sin tener en cuenta las ventajas o desventajas de utilizar otro formato en la infraestructura de desarrollo asociada. Sin embargo, en este trabajo se considera el uso de CSV porque no se utiliza ampliamente para la transferencia de datos, sino más bien para su almacenamiento. Debido a su estructura simple, se propone comparar su rendimiento con el de otros formatos.

Por ello, en este trabajo se propone realizar un análisis de las estrategias de intercambio de datos con el objetivo de diseñar una interfaz de comunicación entre un Chat-Bot en telegram, una herramienta de visualización y diagnóstico (VisTHAA) y un optimizador (MS-DOSS), de forma que un decisor pueda tener una comunicación directa a través de la integración de herramientas de optimización.

1.1 Antecedentes

En esta sección se describen trabajos relacionados con la comunicación entre programas que se han realizado en el ITCM.

Tabla 1 Antecedentes

Tesis	**Aplicaciones conectadas**	**Caso de Prueba**	**Formato de intercambio de datos**
Desarrollo de un protocolo de comunicación para un Framework de apoyo a la toma de decisiones (Sánchez de la Paz, 2016)	Módulos del FAD (Framework de apoyo a la decisión)	Cartera de proyectos	XFDM
Análisis de desempeño usando VisTHAA aplicado a un algoritmo metaheurístico de un framework de optimización (Ponce Nájera, 2021)	VisTHAA y MS-DOSS	Mochila mono objetivo, cartera de proyectos multiobjetivo y DTLZ	JSON
Arquitectura de interacción entre decisores y framework de optimización (Hernández Vicente, 2023)	Chat-bot de Telegram y MS-DOSS	Problemas DTLZ	JSON

En (Sánchez de la Paz, 2016) se realizó un protocolo de comunicación entre distintos módulos del framework de apoyo a la decisión (FAD), para el intercambio de datos utilizaron XFDM el cual es una modificación del formato XML.

En (Ponce Nájera, 2021) se agregó un módulo adicional a VisTHAA para la comunicación con el optimizador, en este módulo se genera la configuración del experimento la cual se procesa y se envía a MS-DOSS para su ejecución. La configuración del experimento, así como el resultado de este se realiza en JSON.

En (Hernández Vicente, 2023) se desarrolló un módulo interactivo entre un chat-bot de telegram y el framework de optimización MS-DOSS para el apoyo de procesos de optimización para la toma de decisiones en presencia de imprecisión e incertidumbre, en este trabajo la arquitectura interactiva permite establecer directamente las preferencias del decisor que pasan del bot al framework que se encuentran en un mismo módulo haciendo uso del formato JSON.

A diferencia del tercer trabajo, en el presente trabajo se agregará la herramienta de diagnóstico VisTHAA, y se hace el análisis del impacto de los formatos esto cuando los datos tienen que pasar a través de las distintas aplicaciones que se conectan, como se observa en la tabla 1,1 los trabajos hacen uso de un solo formato debido a que es el adecuado, a su criterio, para los datos que manejan. Para la realización de los trabajos no se realizaron análisis del uso de distintos tipos de formatos para el intercambio de datos, se optó por utilizar el formato a decisión del autor de acuerdo los datos que se manipulaban.

1.2 Estado del arte

En esta sección se describen los trabajos relacionados con el análisis del impacto del uso de los distintos formatos de intercambio de datos entre aplicaciones.

Tabla 2 Trabajos del estado del arte

Autor	**Caso de prueba**	**Lenguaje de programación**	**Métricas de desempeño utilizadas**	**Base de datos utilizada**	**Formato de intercambio de datos**
(Breje et al., 2018)	API web	PHP con framework Lavarel	Tiempo de respuesta, tamaño de los datos recibidos	Mysql	XML y JSON
(Rianto et al., 2021)	Replicación de base de datos documentales NOSQL	Python 3	Uso de CPU, uso de memoria y de ancho de banda	MongoDB, ArangoDB, RethinkDB	JSON y XML
(Saračević et al., 2016)	Aplicación de autoría	PHP y Flex	Tiempo de ejecución	Mysql	AMF, JSON y XML

En (Breje et al., 2018) los autores realizaron diferentes pruebas de consulta, actualización de una parte de los datos, actualización completa de datos y guardado de los datos, la API que diseñaron recibe datos en XML o JSON, la conversión de los datos la hacen a través de funciones de PHP. JSON presento tiempos de respuesta de 30 a 40% más rápidos que el XML en las distintas peticiones de diversos tamaños, las métricas de desempeño que utilizaron fueron las de tiempo de ejecución (antes y después de validar los datos), el tamaño de los datos recibidos memoria utilizada.

En (Rianto et al., 2021) los autores compararon diferentes métricas desempeño a la hora de realizar la replicación de una base de datos, la base de datos se replicó a distintos gestores como lo son MongoDB, CouchDB y RethinkDB, las métricas usadas fueron el uso de CPU durante el proceso de replicación, memoria RAM utilizada y el tiempo de ejecución, para el cliente usaron el lenguaje Python3. JSON tuvo mejor uso de CPU, pero en las bases de datos CouchDB y RethinkDB no tiene diferencia significativa, mientras que en el tiempo de ejecución XML presentó un tiempo ligeramente menor que JSON.

En (Saračević et al., 2016) los autores realizaron un programa en la aplicación Flex que permite usar código reutilizable para aplicaciones móviles, web, escritorio, el servidor web en php contiene la base de datos mysql en la que hicieron pruebas en AMF, JSON y XML. Las métricas que utilizaron fueron el tiempo (desde la ejecución de la solicitud hasta que se presentan los datos al usuario). Y tamaño del archivo. El autor concluyó que para datos no mayores a 2000 recomienda JSON, seguido de AMF y por último XML,

Como se observa en los artículos presentados si hay diferencia entre el uso de uno u otro formato, pero como se ve en el primer y tercer trabajo cambiando ya sea el tipo de base de datos o a través de que flujo viajan los datos pueden haber resultados opuestos.

En el presente trabajo se conectará tres aplicaciones, dos de las cuales son locales (que no requieren conectarse a internet para funcionar, la diferencia principal es que mientras en los trabajos del estado del arte una aplicación web guarda los datos en una base de datos, en este caso ambas aplicaciones serán invocadas por el Chat-Bot por lo tanto el intercambio de datos será entre el bot y dichas aplicaciones.

1.3 Justificación

Desde hace un tiempo en el ITCM se ha desarrollado una herramienta conocida como VisTHAA (Ponce Nájera, 2021) la cual tiene el objetivo de hacer el análisis y diagnóstico de algoritmos heurísticos. Actualmente se continúa trabajando en VisTHAA y se requiere de una interfaz gráfica para funcionar, además no cuenta con una variedad amplia de algoritmos por lo que se requiere la conexión con el Framework de optimización MS-DOSS.

El framework incluye una mayor variedad de algoritmos, así como una facilidad para agregar nuevos algoritmos. Ambas aplicaciones están separadas y necesitarían otra aplicación que las invoque para que puedan funcionar juntas, esta aplicación que las comunique debe hacerlo de una manera sencilla donde solo pase los parámetros de una aplicación a otra o mande resultados para graficarlos.

Para lograr desarrollar el módulo de comunicación se necesita conocer los diferentes formatos de intercambio. En los trabajos del estado del arte se ha investigado el impacto del uso de distintos formatos de intercambio de datos entre aplicaciones, usualmente cuando se trabaja con aplicaciones web, en las cuales los datos son almacenados en una base de datos, sin embargo, en este caso se tiene dos aplicaciones locales (que no se conectan a internet) que se requieren comunicar entonces los datos no serán almacenados, si no que enviados mediante el módulo de comunicación a la otra aplicación para ser procesados.

Entre los formatos a utilizar, JSON y XML son los más utilizados para la comunicación entre aplicaciones en la web. Esto se debe a que los desarrolladores optan por los formatos ya implementados en el lenguaje que utilizan a la hora de crear estas aplicaciones, ya que les resulta más sencillo; esto sin tener en cuenta las ventajas o desventajas de utilizar otro formato en la infraestructura de desarrollo asociada. Sin embargo, en este trabajo se considera el uso de CSV porque no se utiliza ampliamente para la transferencia de datos, sino más bien para su almacenamiento. Debido a su estructura simple, se propone comparar su rendimiento con el de otros formatos.

La siguiente investigación cubrirá el desarrollo de una interfaz para la comunicación entre un Chat-bot, un framework de optimización y una herramienta de visualización, así como el análisis del impacto de los distintos tipos de intercambio de datos entre estos.

1.4 Objetivos

1.4.1 Objetivo general

- Analizar el impacto de distintos formatos para el intercambio de datos aplicados a la interacción entre un Chat-Bot, un optimizador y una herramienta de visualización y diagnostico para propósito de comunicación de información.

1.4.2 Objetivos específicos

- Caracterización de la información que se va a transferir
- Analizar formatos de intercambio
- Establecer métricas de calidad para el desempeño de los formatos de intercambio de datos en el contexto.
- Analizar y validar el desempeño que los formatos de intercambio en función de las métricas definidas

1.5 Alcances y delimitaciones

- Se considera la creación de un protocolo de comunicación entre las 3 aplicaciones: Chat-Bot, optimizador, herramienta de diagnóstico (Telegram, VisTHAA, MS-DOSS)
- El bot manejará texto y sus respuestas serán texto o imágenes
- Se utilizará NodeJS para la comunicación entre las 3 aplicaciones
- El caso de prueba será la herramienta de toma de decisiones interactiva y con incertidumbre (Hernández Vicente, 2023).
- Se estudiará al menos una necesidad de información entre cada uno de los módulos (Chat-Bot, VisTHAA, MS-DOSS).

1.5 Hipótesis

Como se vio en los trabajos del estado del arte JSON parece ser el mejor formato para intercambiar datos, pero en esos trabajos el intercambio de datos se realiza entre una aplicación web y una base de datos en lugar de varias aplicaciones que reciben distintos formatos, por lo que se propone la siguiente hipótesis

H0: JSON no es el mejor formato de intercambio de datos entre un Chat-Bot, un optimizador y una herramienta de visualización y diagnóstico para propósito de comunicación de información específica

Capítulo 2

Marco teorico

En esta sección se muestran todos los temas relacionados para la realización de este trabajo, así como las definiciones.

2.1 Formatos para el intercambio de datos

En el momento que se requiere conectar dos o más aplicaciones es necesario establecer el formato de intercambio de datos, esto para estandarizar como se leerán o recibirán los datos, existen varios formatos como lo son:

- Texto Plano
- XML
- CSV
- JSON

2.2 XML

En este formato de intercambio su nombre es tomado de sus siglas en ingles Extensible Markup Language. Está compuesto de prólogo y cuerpo del documento, el cual contiene solo un elemento raíz, pero puede contener varios elementos con o sin contenido y los cuales pueden tener atributos.

Los documentos de texto, hojas de cálculo, páginas web y bases de datos son algunos de los campos de aplicación del XML. El metalenguaje aparece como un estándar que estructura el intercambio de información varias plataformas. XML es la evolución lo que se conoce como lenguaje GML por IBM (O'Reilly Media, 1998)

Este formato es utilizado ampliamente en informática sobre todo en desarrollo de sistemas de información: caso particular a mencionar son los sistemas administrativos donde el paso de información entre módulos es importante, y se necesita una estandarización para lograr esta trasferencia, sin embargo el formato XML puede ser usado en otras áreas, como en optimización; donde es posible hacer uso de la estandarización para comunicar los resultados entre distintos módulos u otras aplicaciones para que puedan acceder a la información. En la Figura 1 se puede observar un ejemplo de la estructura de este formato.

2.3 JSON

Es un estándar abierto que utiliza texto plano para codificar información en la estructura atributo: valor. Su nombre proviene de las ingles JavaScript Object Notation y aunque en sus primeros pasos fue considerado parte de JavaScript, JSON es independiente del lenguaje de programación y se puede encontrar disponible para los lenguajes más populares. En un archivo JSON los datos agrupados en conjuntos. Puede tener una estructura jerárquica, pero lo únicos tipos de valores son objetos, vectores, variables y valores (JSON, s.f.). En la Figura 1 se puede observar un ejemplo de la estructura de este formato.

```
{
  "tipo": "Class",
  "nombre": "Persona",
  "operaciones":    ["Consultar","Insertar","Actualizar"],
  "vistas": ["PDF","XML", "cargo; "]
}
```


Figura 1 Ejemplo de la estructura del formato JSON (JSON, s. f.).

2.4 CSV

El formato de archivo CSV (Valores separados por comas) se usa a menudo para intercambiar datos entre aplicaciones dispares. El formato de archivo, tal como se usa en Microsoft Excel, se ha convertido en un pseudo estándar en toda la industria, incluso entre las plataformas que no son de Microsoft.

Normalmente es utilizado para presentar datos en forma de tabla como se muestra en la Figura 2:

Año	Marca	Modelo	Descripción	Precio
1997	Ford	E350	ac, ABS, moon	3000.00
1999	Chevy	Venture	Extended Edition	4900.00
1999	Chevyr	Venture	Extended Edition, Very Large	5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00


```
Año,Marca,Modelo,Descripción,Precio
1997,Ford,E350,"ac, ABS, moon",3000.00
1999,Chevyr,Venture,Extended Edition,4900.00
1999,Chevy,Venture,"Extended Edition, Very Large",5000.00
1996,Jeep,Grand Cherokee,"MUST SELL! air, moon roof, loaded",4799.00
```


Figura 2 Ejemplo de la estructura del formato CSV (Repici, 2002).

Como es el caso con la mayoría de los formatos de intercambio desde XML, los archivos CSV se han convertido en un formato heredado. Las nuevas aplicaciones que deseen incluir un formato de exportación generalmente usarán XML hoy (aunque puede haber excepciones). Sin embargo, en los sistemas heredados (anteriores a XML), los archivos CSV se habían convertido de hecho en un estándar industrial de facto (Repici, 2002).

Dados estos formatos se utilizarán distintas aplicaciones para comprobar su eficiencia en las métricas del tiempo, tamaño, uso de CPU y RAM.

2.5 Telegram

Telegram es una aplicación de mensajería enfocada en la velocidad y seguridad, es súper rápida, simple y gratuita. Puedes usar Telegram en todos tus dispositivos al mismo tiempo. Tus mensajes se sincronizan a la perfección a través de cualquiera de tus teléfonos, tableta o computadoras. Telegram tiene más de 500 millones de usuarios activos mensuales y es una de las 10 apps más descargadas del mundo.

Con Telegram, puedes enviar mensajes, fotos, videos y archivos de cualquier tipo (doc, zip, mp3, etc.), como también crear grupos de hasta 200.000 personas o canales para hacer difusiones a audiencias ilimitadas. Puedes escribir a tus contactos del teléfono y encontrar 21 personas a través de sus nombres de usuario. Como resultado, Telegram es como el SMS y el correo electrónico combinados, y puede satisfacer todas tus necesidades de mensajería personal o de negocios. Además, ofrece llamadas de voz y videollamadas con cifrado end to-end, así como chats de voz en grupos que permiten miles de participantes (*Telegram*, s. f.).

2.5.1 Telegram Bots

Los bots de Telegram son programas de terceros que corren dentro de telegram. Los usuarios pueden interactuar con los bots enviándoles mensajes y comandos. El control de los bots se realiza usando peticiones HTTPS hacia la Bot API de telegram.

¿Qué pueden hacer los bots?

• Recibir notificaciones y noticias personalizadas.

• Integración con otros servicios

• Aceptar Pagos e usuarios de Telegram

• Crear Herramientas personalizadas

• Crear juegos para un jugador y multijugador

• Construir servicios sociales

Aparte de telegram otra herramienta que se necesita integrar a la comunicación entre herramientas es VisTHAA.

2.6 VisTHAA

Esta es una herramienta de diagnóstico para el análisis de algoritmos heurísticos, esto hace posible que los investigadores mejorar las capacidades funcionales mediante módulos que se integran a la arquitectura para favorecer el análisis de algoritmos heurísticos. La Herramienta cuenta con los siguientes módulos disponibles que se pueden observar en la Figura 4 (Cruz-Reyes et al., 2013):

- Entrada de datos y preprocesamiento
- Caracterización de instancias
- Visualización de las instancias y el comportamiento del algoritmo
- Visualización del espacio de búsqueda en tres dimensiones y el análisis de los algoritmos
- Rediseño causal

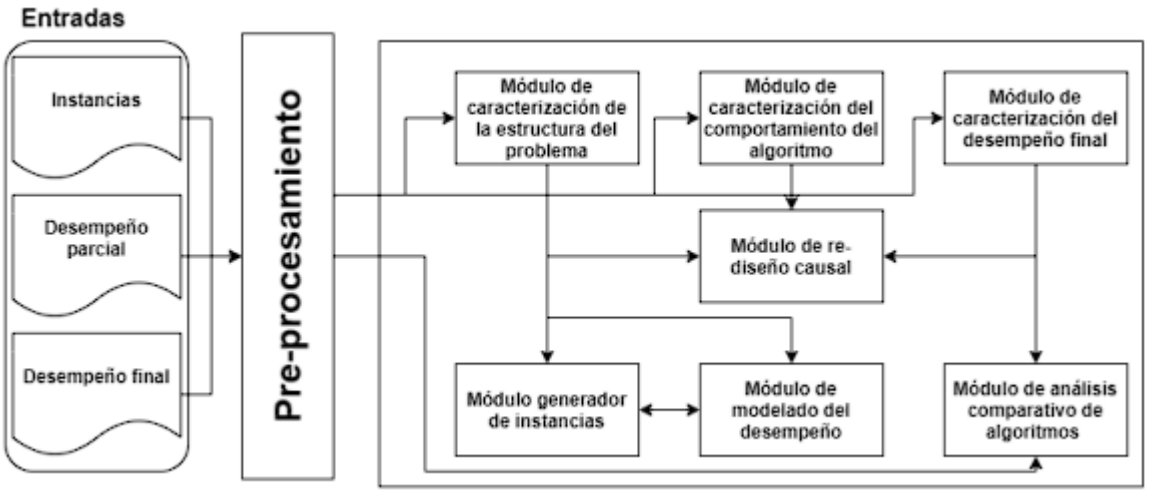


Figura 3 Arquitectura interna de VisTHAA (Ponce Nájera, 2021).

Como se observa en la figura los distintos módulos de VisTHAA ayudan al análisis de algoritmos heurísticos, pero actualmente VisTHAA solo puede ser utilizada a través de su interfaz por lo que se propone la implementación de una API para el uso de los módulos sin invocar a la interfaz gráfica.

Aparte de telegram y VisTHAA, otra herramienta que se necesita integrar a la comunicación entre herramientas es el framework de optimización MS-DOSS.

2.7 Framework de Optimización

En esta sección se describen los framework, haciendo énfasis en los framework de optimización ya que el MS-DOSS es uno de ellos.

2.7.1 Framework

Muente en el 2020 (Muente, 2020) describió el framework como un conjunto de archivos y directorios que facilitan la creación de aplicaciones, ya que incorporan funcionalidades ya desarrolladas y probadas, implementadas en un determinado lenguaje de programación.

El objetivo principal de todo framework es facilitar las cosas a la hora de desarrollar una aplicación, haciendo que se centren en el verdadero problema y se olviden de implementar funcionalidades que son de uso común como puede ser el registro de un usuario, establecer conexión con la base de datos, manejo de sesiones de usuario o el almacenamiento en base de datos de contenido cacheado.

Los framework permiten entregar un proyecto en menos tiempo y con un código más limpio, cuya eficacia ya ha sido comprobada. A partir del framework los programadores pueden complementar y/o modificar la estructura base para entregar el software o la aplicación que cumpla los objetivos requeridos.

Ventajas de utilizar un framework para el desarrollo de software:

- El programador ahorra tiempo ya que dispone ya del esqueleto sobre el que desarrollar una aplicación.
- Facilita los desarrollos colaborativos, al dejar definidos unos estándares de programación.
- Al estar ampliamente extendido, es más fácil encontrar herramientas, módulos e información para utilizarlo.

- Proporciona mayor seguridad, al tener gran parte de las potenciales vulnerabilidades resueltas.
- Normalmente existe una comunidad detrás, un conjunto de desarrolladores que pueden ayudar a responder consultas.

Desventajas de utilizar un framework:

- Se requiere de cierto tiempo de aprendizaje.
- Puede existir exceso de líneas de código.
- Limitación en cuanto a las modificaciones que se pueden hacer sobre el framework.

Tipos de framework:

- Para aplicaciones web. Son aquellos framework que se utilizan específicamente para la creación de proyectos online.
- Para aplicaciones en general. Permite complementar la estructura de una aplicación para un sistema operativo.
- Para tecnología de JavaScript Asíncrono con XML (AJAX). La tecnología AJAX permite que el usuario haga solicitaciones al servidor sin que sea necesario recargar una página después de cada nueva solicitud.
- De gestión de contenidos. facilita la programación de aplicaciones de un Sistema de Gestión de Contenidos, popularmente conocido como CMS, por ejemplo, WordPress.
- De Multimedia. facilita el trabajo de los programadores que trabajan con video, audio e imagen y colabora con la creación de las aplicaciones multimedia en general, pudiendo servir para proyectos más complejos, como videoconferencias y conversores de medios.

Estos softwares que se van a conectar mediante la herramienta de comunicación tienen el objetivo de resolver problemas de optimización.

2.7.2 Optimización

Velázquez en el 2010 (Velázquez, s. f.) escribió un libro de modelos matemáticos de optimización del cual se obtienen todas las siguientes definiciones. La investigación operativa se puede definir como la aplicación de métodos científicos en la mejora de la efectividad en las operaciones, decisiones y gestión. o como la ciencia de aplicar los recursos disponibles para conseguir la satisfacción óptima de un objetivo específico deseado.

La principal característica consiste en construir un modelo científico del sistema del cual se pueden predecir y comparar los resultados de diversas estrategias, decisiones, incorporando medidas del azar y del riesgo El objetivo es ayudar a los responsables a determinar su política y actuaciones en forma científica.

Los profesionales de la investigación operativa colaboran con los decisores en el diseño y mejora de las operaciones y decisiones, resuelven problemas y ayudan en las funciones de gestión, planificación o predicción, aportan conocimiento y ayuda en la toma de decisiones. Aplican las técnicas científicas más adecuadas seleccionadas de la matemática, ingeniería o cualquier ciencia social o de administración de empresas. Su trabajo normalmente consiste en recoger y analizar datos, desarrollar y probar modelos matemáticos, proponer soluciones o recomendaciones, interpretar la información y, en definitiva, ayudar a implantar acciones de mejora. Como resultado desarrollan e implantan aplicaciones informáticas, sistemas, servicios técnicos o productos.

Tipos de optimización:

- Lineal.
- No lineal.
- Entera.
- Estocástica.
- Multiobjetivo.

La optimización es una parte relevante dentro de la investigación operativa. Tuvo un progreso algorítmico inicial muy rápido. Los problemas de optimización se componen generalmente de estos tres ingredientes:

Función objetivo.

Es la medida cuantitativa del funcionamiento del sistema que se desea optimizar (maximizar o minimizar).

Ejemplos de función objetivo: la minimización de los costes variables de operación de un sistema eléctrico, la maximización de los beneficios netos de venta de ciertos productos, la minimización del cuadrado de las desviaciones con respecto a unos valores observados, la minimización del material utilizado en la fabricación de un producto, etc.

Variables

Representan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Se pueden clasificar en variables independientes, principales o de control y variables dependientes, aunque matemáticamente son iguales.

Restricciones

Representan el conjunto de relaciones (expresadas mediante ecuaciones e inecuaciones) que ciertas variables están obligadas a satisfacer. Por ejemplo, las potencias máxima y mínima de operación de un grupo de generación, la capacidad de producción de la fábrica para los diferentes productos, las dimensiones del material bruto del producto, etc.

Resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables para hacer óptima la función objetivo satisfaciendo el conjunto de restricciones.

Los métodos de optimización se pueden clasificar en:

Clásicos.

Metaheurísticos. Aparecieron ligados a lo que se denominó inteligencia artificial e imitan fenómenos sencillos observados en la naturaleza.

Como menciona (Ramos, s. f.) resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables para hacer óptima una función objetivo y satisfacer el conjunto de restricciones. En los cuales existen algunos tipos de problemas de optimización que se alteran medianamente este esquema:

- Sistemas de ecuaciones lineales – no lineales en el cual no existe una función objetivo con tal y únicamente interesa encontrar una solución factible a un problema de un conjunto de restricciones.

- Optimización sin restricciones en el que se trata de encontrar un conjunto de valores que indican el mínimo/máximo de una función.
- Optimización multiobjetivo en el cual existen más de una función objetivo y el problema que se plantea es como tratar varias funciones objetivo al mismo tiempo tomando en cuenta que el valor objetivo de una función no es el mismo para otra.

Los métodos de optimización se pueden clasificar como métodos clásicos los cuales habitualmente se explican en la literatura y los métodos metaheurísticos que están ligados a lo que se denominó inteligencia artificial. Dentro de los métodos clásicos podemos encontrar la optimización lineal, lineal entera mixta, dinámica, estocástica, no lineal etc.

En el segundo se incluyen los algoritmos evolutivos (genéticos entre otros), búsquedas heurísticas como el método tabú, búsqueda aleatoria, etc. De manera general y aproximada se puede decir que los métodos clásicos garantizan encontrar el óptimo local y los metaheurísticos dadas las mecánicas específicas que tienen les permite con frecuencia alcanzar los óptimos globales.

Las restricciones que definen las cotas del problema dividen el espacio de búsqueda en dos las cuales son:

- Soluciones factibles: Son aquellos elementos en el espacio de búsqueda que cumplen con las restricciones.
- Soluciones no factibles: Son aquellas soluciones que no toman parte del dominio de la función porque viola al menos una restricción del problema

Una herramienta para diseñar estrategias que solucionan problemas de optimización multiobjetivo es el framework de optimización MS-DOSS.

2.7.3 Framework MS-DOSS

Este permite la declaración de problemas de optimización, así como la definición y configuración de algoritmos solucionadores a un problema dado (Ponce Nájera, 2021).

Tabla 3 Módulos del MS-DOSS

Características de los módulos de optimización	
Módulo	**Descripción**
Validación **1**	Mediante el uso de pruebas estadísticas como Friedman y Wilcoxon, permitirá seleccionar las mejores soluciones creadas por los algoritmos, en función de cada problema. Además, mediante la generalización algunos operadores estadísticos como la media, varianza y desviación estándar, facilitara la creación de nuevas pruebas estadísticas. Este módulo está compuesto por las clases: *Validate, StatisticsSet, Test_Friedman, Test_Wilcoxon, Operator_Statistic, Mean, Variance y Standard_Desviation.*
Preferencia **2**	Este módulo aborda un sistema relacional de preferencias compuesto de varias relaciones binarias [Roy, 1996] como: Indiferencia, Preferencia estricta, Preferencia estricta, Preferencia débil, Incomparabilidad, K-preferencia y No preferencia. Este módulo ayudará a implementar diferentes técnicas pertenecientes a MCDA; el cual se compone por las clases: *Preference* y *ELECTRE III.*
Reporte **3**	Este módulo presenta los resultados obtenidos del modulo "Validación", de tal manera que puedan ser revisados y comparados para su posterior análisis. Este módulo está compuesto por la clase: *Report.*
Instancia **4**	Este módulo está diseñado para contener la instancia de cualquier problema, de tal manera al generalizar su información, pueda ser abordado por el módulo "Problema"; sin importar el formato que utilicen los autores de cada instancia. Actualmente este módulo está compuesto por clases: *Instance, JSSP_Inst* y *Thompson_JSSP*.
Problema **5**	Este módulo permite generalizar distintos problemas de optimización enfatizando sus características comunes. Actualmente este módulo este compuesto por las clases: *Problem, JSSP* y *PPP*.
Algoritmo **6**	Este módulo permite manipular la información generada por el módulo "Problema", según el comportamiento de cada algoritmo. Este permite generalizar distintos algoritmos, enfatizando sus características comunes, facilitando la creación y extensión de sus clases. Este modulo este compuesto por las clases: *Algorithm, Genetic, SimulatedAnnealing* y *NOSGA.*

En la tabla 3 se describen las características de los módulos del framework de optimización, el cual es parte fundamental en la construcción del proyecto.

Para la realización de este proyecto se tiene una herramienta que hace uso del framework de optimización MS-DOSS la cual es definida en la siguiente sección.

Características de los módulos de optimización (Continuación)	
Módulo	**Descripción**
Operadores **7**	Este módulo contiene un conjunto de operadores en el área de optimización, como p. e, Cruza, Mutación, Selección, etc. Estos son necesarios para llevar a cabo la ejecución de múltiples algoritmos como p.e. El Genético, Recosido Simulado y NOSGA (todos estos dentro del módulo "Algoritmo"). Este módulo está compuesto por las clases: Operator, Crossover, Mutation, Selection, etc.
Solución **8**	Este módulo permite generalizar las soluciones creadas por el módulo "Algoritmo"; que, de otra manera, se tendría que crear tantas clases de soluciones como existan algoritmos en el Framework. Este módulo está compuesto por las clases: Solution, SolutionSet.
Diseño experimental **9**	Este módulo es uno de los más importantes de este Framework, ya que permite realizar una experimentación sencilla y cómoda. Además, también permite configurar fácilmente los parámetros pertenecientes a módulo de optimización (p. e. "Problema", "Algoritmo", etc.). De esta manera evitemos la configuración directa a esos módulos. Actualmente este módulo está compuesto por las clases: Parameter, Parameters, Exp_Desing, Gen_Exp, SA_Exp y NOSGA_Exp e Instance_Exp.

2.8 Herramienta de toma de decisiones interactiva y con incertidumbre

Para comprobar el desempeño de cada uno de los formatos es necesaria la comunicación entre aplicaciones, en este caso como caso de estudio se tiene una herramienta interactiva para el apoyo a la toma de decisiones en procesos de optimización en presencia de imprecisión e incertidumbre (Hernández Vicente, 2023). en esta aplicación la comunicación es directa entre el chat bot y el framework MS-DOSS.

En esta sección se presenta la arquitectura de integración de los módulos de la herramienta que se tomó como caso de estudio,

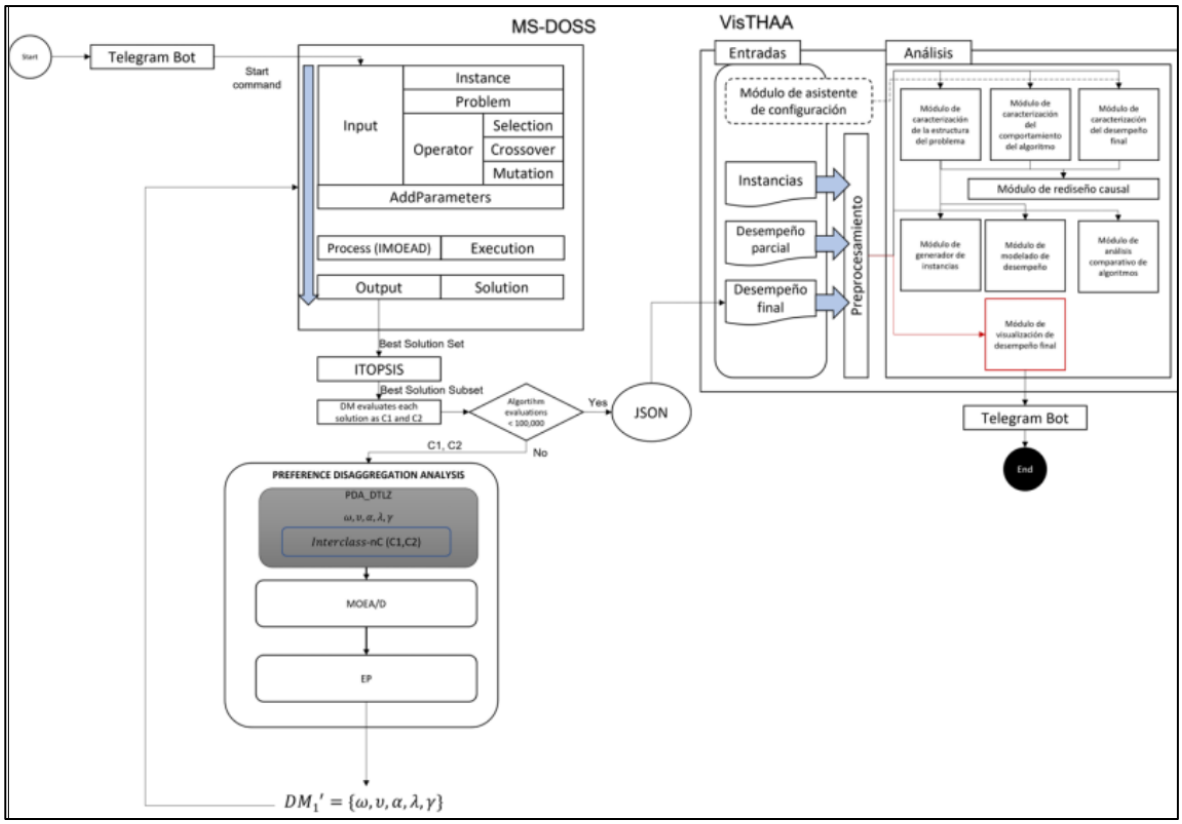


Figura 4. Arquitectura de integración de la herramienta de caso de estudio

En la figura 4 se puede observar que el DM inicia el proceso interactivo con el bot utilizando el comando principal.

Posteriormente, el bot ejecuta un microservicio del framework de optimización MS-DOSS para resolver DTLZ utilizando MOEA/D/ O (La primera vez se utilizan valores extremos en los parámetros de configuración del algoritmo).

A partir del paso anterior, se obtienen N soluciones, que se ordenarán utilizando el método TOPSIS para elegir las 10 mejores. Luego, cada solución del paso anterior se presenta al DM para que pueda seleccionar cuáles le gustan (C2) y cuáles no (C1).

Si el número de evaluaciones del algoritmo principal es inferior a 100,000, entonces el PDA se resuelve utilizando MOEA/D y el conjunto de referencia evaluado por el DM en el paso anterior, obteniendo así un nuevo conjunto de parámetros que serán utilizados por el algoritmo principal

(MOEA/D/O) en la siguiente iteración.

2.9 MOEAD/O

A continuación, se define el algoritmo del MOEAD/O que es una modificación del MOEA/D el cual es utilizado en la herramienta anteriormente descrita para la solución del problema DTLZ.

Algorithm 1. MOEA/D/O

Input:

m= objective number

N= scalar subproblems number

$\varpi=\{\varpi_1, \varpi_2, \ldots, \varpi_N\}$ = set of N weight vectors

$T=N/10$, neigborhood size of each weight vector

DM= (w, v, λ, β), value system of a DM

R_k = A chosen preference relation

Output:

$x=\{x_1, x_2, \ldots, x_N\}$ = last generation of solution associated to the weight vectors.

1. $(x,z,FV,B(i)) \leftarrow$ Initialization(λ,N,m,T)
2. **do**
3. **for** i = 1 to N do
4. $\{y_1, y_2\} \leftarrow$ Reproduction(x,B(i),T)
5. UpdateZ(z, $\{y_1, y_2\}$) // z: for each j =1, …, m, if $z_j < f_j(y')$ then set $z_j = f_j$ (y'), y' $\in \{y_1, y_2\}$
6. **UpdateNeighborhood(x,$B(i)$, λ_i,FV,$\{y_1, y_2\}$, R)** // for each $j \in B(i)$ and $y \in \{y_1, y_2\}$, // if $g^{te}(y \mid \varpi_i,z) \leq g^{te}(x_j \mid \varpi_i,z)$ and $x\ R_k^{DM}\ y$ set $x_j = y$ and $FV_j = F(y)$
7. **end for**
8. **while** Stopping Criterion unsatisfied //when numEvaluations < maxEvaluations

Figura 5 Pseudocódigo del algoritmo MOEA/D/O.

Esta variante del algoritmo MOEA/D, combina intervalos y modelos outranking para representar la variabilidad presente en las preferencias del DM y así guiar el proceso de búsqueda (Fernández et al., 2022).

Esta variante al integrar modelos de outranking durante el proceso de evolución, permite manejar las preferencias del DM ya que se considera una dominancia parcial en lugar de la estricta, los cuales son especialmente útiles para las preferencias subjetivas.

En la figura 5 se puede observar que los nuevos parámetros que recibe el algoritmo son el sistema de evaluación del DM y la relación outranking. También en la línea 6 se observa que en el proceso de actualización de la solución se debe considerar la condición Tchebycheff y la relación de preferencia especificada.

Haciendo uso de este algoritmo se resolverá el problema DTLZ con 5 objetivos y 9 variables definido a continuación

2.10 Método TOPSIS

TOPSIS (Technique of Order Preference Similarity to the Ideal Solution) es un método para el análisis de decisiones multicriterio, desarrollado por Yoon en 1981 (Rodríguez, 2020).

Está basado en la idea de que la solución deseable debería de tener la distancia geométrica más corta a la solución ideal y la distancia geométrica más prolongada a la solución anti-ideal.

La alternativa ideal es aquella que cuenta con una distancia geométrica más cercana al frente de Pareto, mientras que la alternativa anti-ideal tiene una distancia muy lejana al frente de Pareto.

2.10.1 ITOPSIS

A diferencia del método topsis normal esta versión los pesos de los criterios se presentan en una matriz de decisión de intervalos para incorporar el uso de incertidumbre en los datos.

Los pasos utilizados son los siguientes:

$$\otimes D = \begin{bmatrix} \otimes x_{11} & \cdots & \otimes x_{1m} \\ \vdots & \ddots & \vdots \\ \otimes x_{n1} & \cdots & \otimes x_{nm} \end{bmatrix}; i = 1, \dots, n; j = 1 \dots, m$$

Donde:

$\otimes x_{ij} =$ Expresa las evaluaciones de intervalos de la alternativa i (cartera) con respecto a j (Objetivos).

$n =$ número de carteras.

$m =$ número de objetivos.

El siguiente paso consiste en la construcción de la matriz de decisión normalizada. Para ello es necesario la utilización de la formula siguiente. Una vez obtenida la matriz de decisión normalizada se procede a determinar la alternativa ideal A^+ y la anti-ideal A^- de cada objetivo.

Para la obtención de la alternativa A^+ es necesario la aplicación de la formula siguiente:

$$A^+ = \{(\max_i r_{ij} | j \in J), | i \in n\}$$

Y para el cálculo de la alternativa se emplea la siguiente ecuación:

$$A^- = \{(\min_i r_{ij} | j \in J), | i \in n\}$$

Al realizar el cálculo de las dos alternativas se tiene como resultado dos vectores, una para cada alternativa. Obteniendo las siguientes ecuaciones:

$$A^+ = [r_1^+, r_2^+, \dots, r_m^+]$$

$$A^- = [r_1^-, r_2^-, \dots, r_m^-]$$

Donde:

$$\otimes r_{ij} = \frac{\otimes x_{ij}}{max_i(\overline{x}_{ij})|} = \left(\frac{\underline{x}_{ij}}{max_i(\overline{x}_{ij})}; \frac{\overline{x}_{ij}}{max_i(\overline{x}_{ij})}\right)$$

$n =$ número de carteras.

$m =$ número de objetivos.

Como siguiente procedimiento es el calcular la distancia de separación a la alternativa ideal y anti-ideal, Requiriendo la utilización de la distancia Euclidiana. La separación de cada alternativa i (Cartera) a la alternativa ideal A^+, es obtenida a través de la siguiente ecuación:

$$d_i^+ = \sqrt{\frac{1}{2}\sum_{j=1}^{m} w_j \left[\left|r_j^+ - \underline{r}_{ij}\right|^2 + \left|r_j^+ - \overline{r}_{ij}\right|^2\right]}$$

Donde:

$r =$ número de carteras.

$m =$ número de objetivos.

$w_j =$ pesos del beneficio j.

La separación de cada alternativa i (Cartera) a la alternativa anti-ideal A^+, es obtenida a través de la siguiente ecuación:

Como último proceso del algoritmo es el cálculo de la cercanía relativa a la solución ideal. Haciendo uso de las distancias y de la siguiente ecuación:

$$C_i^+ = \frac{d_i^-}{d_i^+ + d_i^-}$$

Donde:

$0 \leq C_i^+ \leq 1$. Mientras más cercano a 1 es el valor, mejor será la evaluación de la alternativa.

2.11 Análisis de Desagregación de Preferencias

La desagregación de preferencias consiste en el análisis de las preferencias globales del tomador de decisiones (DM) para deducir la importancia relativa del criterio de evaluación y así desarrollar el modelo de preferencia correspondiente a las preferencias globales (Doumpos & Zopounidis, 2004).

El PDA es en sí un problema a resolver, el cual consiste en estimar una función que sea consistente con las preferencias subjetivas conocidas por el DM (Doumpos & Zopounidis, 2004).

$$d_i^- = \sqrt{\frac{1}{2}\sum_{j=1}^{m} w_j \left[\left| r_j^- - \underline{r}_{ij} \right|^2 + \left| r_j^- - \overline{r}_{ij} \right|^2 \right]}$$

Para resolver el PDA se necesita un conjunto de soluciones de referencia (RS) previamente evaluadas por el DM como soluciones malas (C1) o soluciones buenas (C2).

Posteriormente MOEA/D Debe resolver el problema PDA(RS) para estimar un nuevo conjunto de parámetros preferencial (pesos, vetos, credibility, majority, dominance) del Modelo de Outranking $nCinf = \{w, v, \alpha, \lambda, \gamma\}$.

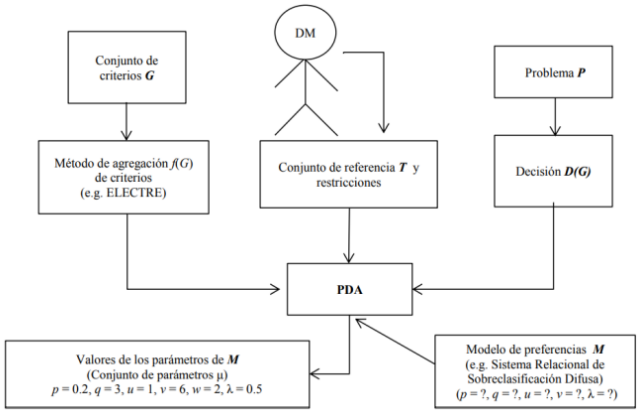


Figura 6 Elementos involucrados en la resolución del PDA (Hernández Vicente, 2023).

2.12 Problemas Estándar

De acuerdo con (Li, 2015). Los problemas de prueba DTLZ fueron construidos por Deb en 2005, y constan de una serie de funciones de prueba comúnmente usados para evaluar y comparar el rendimiento de algoritmos evolutivos (en este caso el MOEA/D/O), lo cual también contribuye a la mejora de estos. Hay un total de 9 funciones de la suite DTLZ y cada una tiene su propia naturaleza de manera que se pueda probar una habilidad especifica de los algoritmos evolutivos.

2.13 DTLZ1

Este está compuesto por *M* objetivos y es un problema de prueba simple debido a que su límite óptimo es un hiperplano lineal, el cual se puede expresar con la siguiente ecuación:

$$\begin{cases} \min f_1(x) = \frac{1}{2} x_1 x_2 \dots x_{M-1} [1 + g(X_m)] \\ \min f_2(x) = \frac{1}{2} x_1 x_2 \dots (1 - x_{M-1}) [1 + g(X_m)] \\ \dots \\ \min f_{M-1}(x) = \frac{1}{2} x_1 (1 - x_2) [1 + g(X_m)] \\ \min f_M(x) = \frac{1}{2(1 - x_1)[1 + g(X_m)]} \end{cases}$$

$$g(X_m) = 100 \left\{ |X_m| + \sum_{x_i \in X_m} (x_i - 0.5)^2 - \cos [20\pi (x_i - 0.5)] \right\}$$

Donde X_m es la última *N - M + 1* variable de decisión en el vector de decisión x y $|X_m| = N - M + 1$. Para cada *i* {1,2, …, *n*}, se satisface $0 \leq x_i \leq 1$. El límite óptimo objetivo es $\sum_{i=1}^{M} f_i = 0.5$.

Dado este problema haciendo uso del algoritmo MOEA/D/O se generarán resultados, que serán utilizados para el intercambio de datos entre las distintas aplicaciones.

Una vez hecha la experimentación los datos recopilados de esta se compararán entre cada uno de los formatos en las distintas métricas establecidas para observar cual tiene mejor rendimiento, y una vez teniendo estos datos es necesario comprobar si estas diferencias son significativas o no para lo cual se hará un análisis estadístico de los datos.

2.14 Incertidumbre

Los problemas de optimización multiobjetivo pueden presentar conflictos entre los objetivos, el DM puede expresar cada objetivo, como un intervalo tomando en cuenta la imprecisión, esto es, $f_j(x) = \left[\underline{f_j}(x), \overline{f_j}(x)\right]$. Cada elemento del conjunto *X* es tratado como un vector de intervalos $\vec{f}(x)$.

$$max\ F(x) = (f_1, f_2, \dots, f_m) \quad (1)$$

$$Sujeto\ a:\ x \in \Omega$$

La incertidumbre proviene de un conocimiento incompleto. El término se puede emplear en diferentes situaciones; es comúnmente utilizado en los campos de la estadística y la economía, donde determinadas circunstancias hacen imposible proporcionar un diagnóstico preciso de que sucederá en el futuro (Balderas, 2016).

2.15 Análisis estadístico

Según (Anderson et al., 2019), el análisis estadístico es el proceso de examinar, resumir y analizar datos con el objetivo de extraer información significativa, revelar patrones, tendencias, y tomar decisiones informadas en una amplia variedad de campos, desde la investigación científica hasta la toma de decisiones en negocios y política.

Una vez recopilados los datos se hizo uso de la plataforma STAC para estimar la prueba estadística que mejor se ajusta a los datos, el proceso que realiza STAC toma en cuenta los siguientes datos:

- El número de grupos disponibles
- El número de muestras por grupo
- Emparejamiento entre los grupos
- La normalidad de cada grupo (comprobada mediante la prueba de Shapiro-Wilks con alfa de 0.1)
- La homocedasticidad entre grupos (comprobada mediante la prueba de Leven con alfa de 0.1)

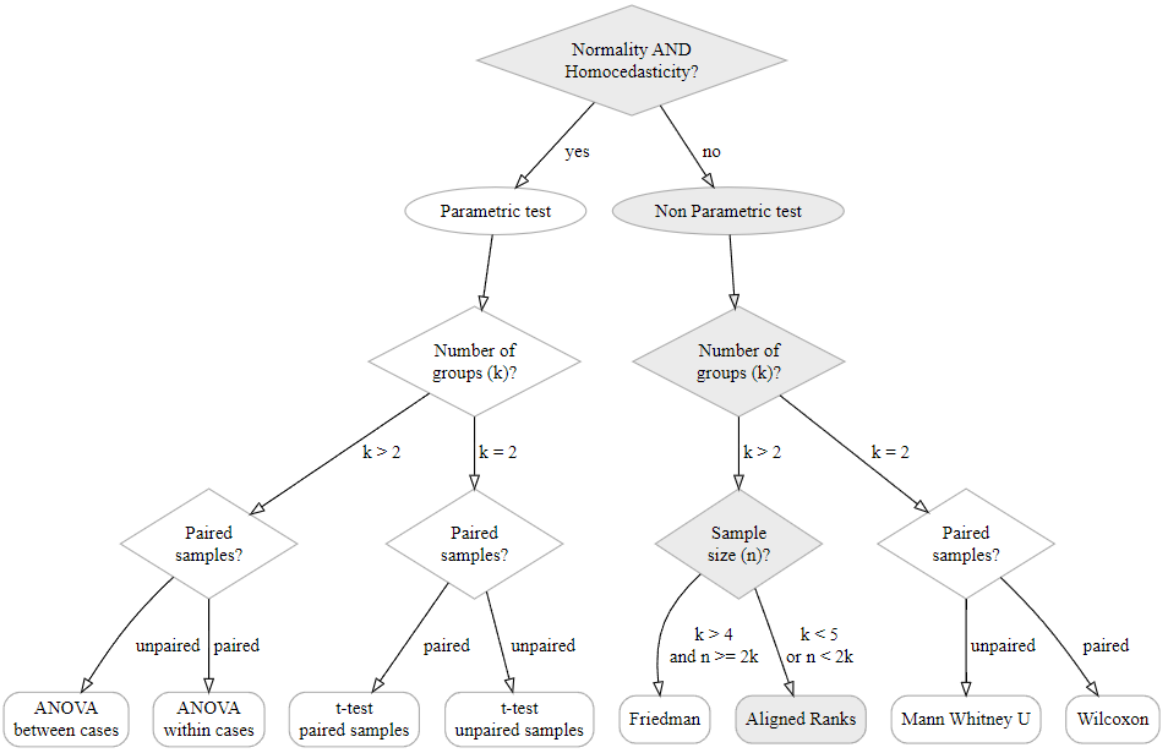


Figura 7 Proceso de decisión usado por STAC para la elección de la prueba estadística

Como se observa en la figura 7 STAC recomienda el uso de una prueba no paramétrica en este caso la prueba de rangos alineados de Friedman.

2.16 Pruebas no paramétricas

En los casos en que los datos no cumplen con los supuestos de normalidad y homogeneidad de varianzas requeridos por las pruebas paramétricas, se debe recurrir al uso de pruebas no paramétricas que son apropiadas cuando se trabaja con datos que no siguen una distribución normal o cuando se tienen muestras pequeñas.

Como se observó en la figura 6 los datos recopilados cumplen con una distribución normal, pero no con la homocedasticidad por lo que se recomienda el uso de una prueba no paramétrica, como el número de grupos es bajo, 3 en este caso (CSV, JSON y XML), se recomienda el uso de la prueba

de Rangos alineados de Friedman.

2.16.1 Prueba de rangos alineados de Friedman

(Liu & Xu, 2022) lo definen de la siguiente manera:

La prueba de Friedman de rangos alineados calcula el valor medio de rendimiento de todos los algoritmos en cada conjunto de datos y, a continuación, resta el valor medio correspondiente de los resultados de todos los algoritmos para obtener la diferencia. La clasificación de los algoritmos se obtiene con respecto a la diferencia en todos estos N conjuntos de datos, donde el algoritmo con mayor diferencia ocupa el puesto 1, y el de menor diferencia ocupa el puesto N × K.

Por último, se calcula el valor del estadístico de la ecuación en función de los valores de clasificación obtenidos:

$$T = \frac{(K-1)\left[\sum_{j=1}^{K}\widehat{R}_{\cdot j}^{2} - \left(KN^{2}/4\right)(KN+1)^{2}\right]}{\left\{\left[KN(KN+1)(2KN+1)\right]/6\right\} - (1/K)\sum_{i=1}^{N}\widehat{R}_{i\cdot}^{2}},$$

donde $R_{i.}$ es la suma de las clasificaciones de todos los algoritmos en el i-ésimo conjunto de datos y $R_{.j}$ es la suma de las clasificaciones del j-ésimo algoritmo en todos los conjuntos de datos.

El estadístico T mostrado obedece a una distribución chi-cuadrado con K-1 grados de libertad. El valor crítico para el nivel de significación especificado puede obtenerse consultando la tabla de distribución chi-cuadrado. Al comparar el valor de T con este valor crítico, se rechaza la hipótesis nula si el valor de T es mayor que el valor crítico, y viceversa.

2.16.2 Prueba Posthoc

Como lo definen (Liu & Xu, 2022) la prueba de Friedman puede determinar si existen diferencias significativas entre los algoritmos considerados en los experimentos. Sin embargo, no puede identificar cuál es significativamente mejor que los demás. Por lo tanto, una vez rechazada la hipótesis nula por estas pruebas de comparación múltiple, suele ser necesario adoptar un procedimiento post hoc de comparación por pares.

Las pruebas post hoc se utilizan para descubrir diferencias específicas entre tres o más medias de grupos cuando una prueba F de análisis de varianza es significativa. Debido a que la prueba F es "ómnibus", simplemente indicará a los investigadores que existe una diferencia entre los grupos, pero no entre qué grupos específicamente. Las pruebas post hoc permiten a los investigadores localizar esas diferencias específicas y se calculan sólo si la prueba F general es significativa. Si la prueba F general no es significativa, entonces no es necesario que el investigador explore diferencias específicas (Wiedmaier, 2017).

Por lo tanto, al rechazarse la hipótesis nula al hacer la prueba de Friedman es necesario realizar este tipo de prueba para identificar las diferencias especificas entre los grupos.

Capítulo 3

Propuesta de solución

En esta sección se muestra la arquitectura de solución propuesta con la función de cada uno de sus módulos a su vez se muestran los cambios realizados en la arquitectura original para la realización de este trabajo.
En la siguiente figura se encuentra la arquitectura utilizada para este trabajo.

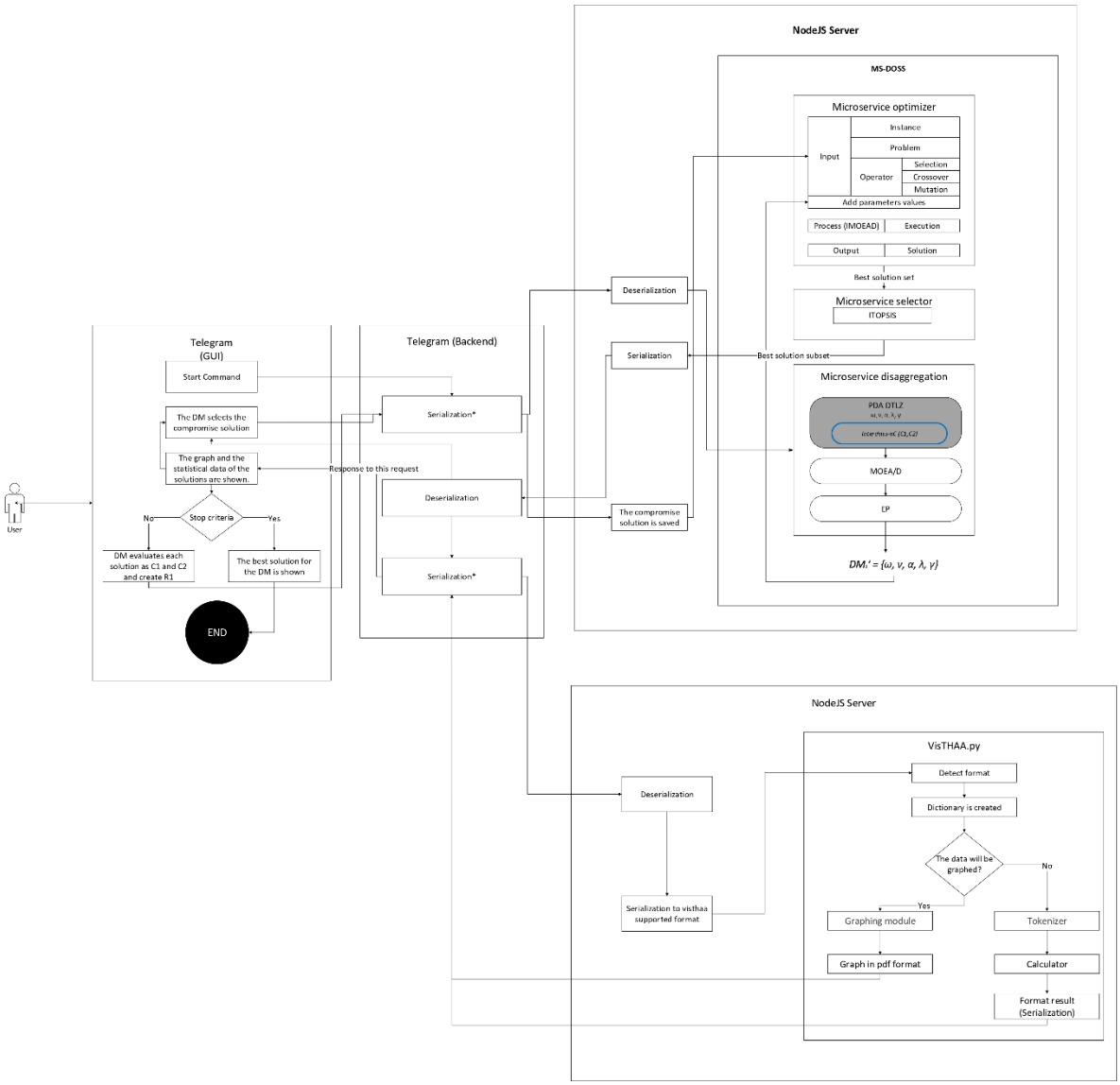


Figura 8 Arquitectura de solución propuesta.

Como se observa en la figura 10, la arquitectura de solución propuesta en este trabajo se compone de 3 módulos, el módulo de telegram, el framework MS-DOSS y VisTHAA. Además, se llevó a cabo un análisis de formatos de comunicación con el objetivo de integrarlos y actualizar la

comunicación entre los módulos.

En el trabajo de (Hernández Vicente, 2023) se trabajó con los mismos módulos, pero no había una interfaz de comunicación que transformara los datos para ser compartidos entre módulos. Anteriormente el módulo de MS-DOSS y el chatbot estaban combinados y no necesitaban una interfaz de comunicación, pero en esta nueva arquitectura propuesta dichos módulos están separados por lo cual es necesario la interfaz de comunicación que transformara los datos para ser compartidos, otra aportación de este trabajo es la incorporación de los formatos CSV y XML.

El módulo de telegram en este trabajo de investigación se propone dividirlo en dos procesos, la primera es la interfaz gráfica donde el usuario interacciona con la herramienta la cual le pasa la información a la interfaz de comunicación para iniciar el proceso y ser enviada al Backend el cual se encuentra en una aplicación de NodeJS que se encarga de recibir las peticiones

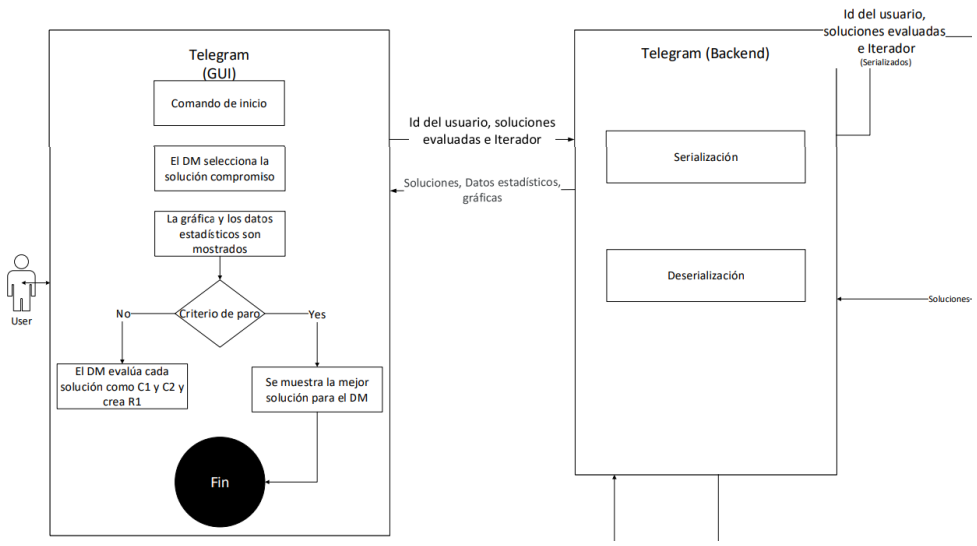


Figura 9 Arquitectura de solución (Módulos de Telegram)

En el módulo de interfaz gráfica el usuario recibirá las soluciones que deberá indicar si le gustan o no, mediante un ciclo el Backend enviara estas soluciones calificadas por DM hacia el framework para generar nuevas soluciones en base a las preferencias, o a VisTHAA para la generación de las gráficas y de los datos estadísticos (Mediana, Varianza, Desviación estándar, etc.).

El módulo del Backend es el que controlas las peticiones recibidas por telegram, este es el módulo que se encarga de comunicarse con el Framework de optimización y la herramienta de análisis y diagnóstico, este módulo se encarga de serializar los datos para su envió a una de las aplicaciones, o deserialización de los datos para ser mostrados al usuario.

3.1 Caracterización de la información que se va a transferir

Para el desarrollo de esta actividad fue necesario separar el módulo de comunicación entre el bot y el framework MS-DOSS, inicialmente la comunicación entre estos dos módulos era directa y el bot solo se podía comunicar con el framework MS-DOSS ya que esta comunicación era fija. En la nueva arquitectura (mostrada en el siguiente tema) el framework MS-DOSS se transforma en un microservicio, entonces la comunicación ya no es fija esto significa que pueda conectarse con otros microservicios en caso de que fuera necesario. Además, se añadió el uso de los formatos CSV y XML en la comunicación entre los módulos.

Recordemos que en el trabajo de (Hernández Vicente, 2022) la interacción entre el bot y el framework consta de tres etapas, la inicial en la cual se solicitan las soluciones aleatorias (donde se obtiene la solución compromiso), la segunda donde se obtienen soluciones por MOEAD, y la tercera en la que dada las soluciones anteriores y el conjunto de referencia (donde el decisor indica que solución le gusto o no) se ejecuta el PDA que obtiene los pesos, vetos y umbrales para establecer las preferencias del DM y poder ejecutar nuevamente la generación de las soluciones con estos valores.

A su vez VisTHAA se encapsula en su propio microservicio, para el propósito de este trabajo es necesaria la transferencia de datos por lo que ahora con esta arquitectura ocurre esta transferencia debido a que antes la comunicación era directa. La diferencia importante con la herramienta original es la comunicación a través de microservicios y el uso de los formatos de intercambio de datos, que permitirán tener flexibilidad en la comunicación y usar el mejor el mejor formato en función del tiempo y la capacidad.

Los datos que se transfieren entre las aplicaciones son los siguientes:

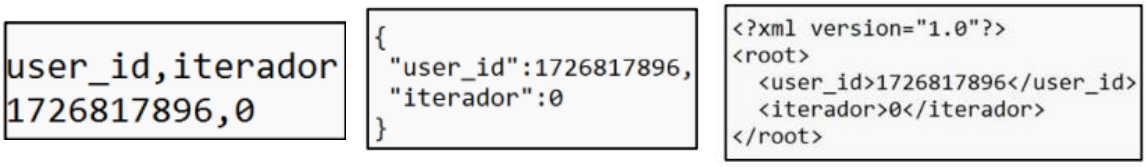
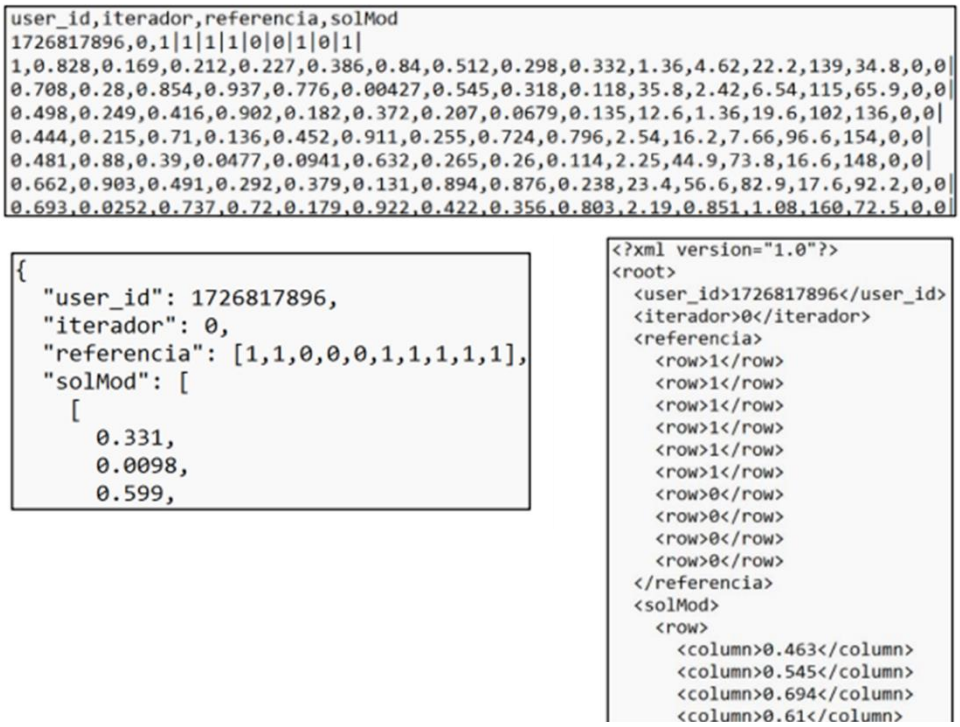


Figura 10 Datos iniciales recibidos por el MS-DOSS en formato CSV, JSON y XML



```
user_id,iterador,referencia,solMod
1726817896,0,1|1|1|1|0|0|1|0|1|
1,0.828,0.169,0.212,0.227,0.386,0.84,0.512,0.298,0.332,1.36,4.62,22.2,139,34.8,0,0|
0.708,0.28,0.854,0.937,0.776,0.00427,0.545,0.318,0.118,35.8,2.42,6.54,115,65.9,0,0|
0.498,0.249,0.416,0.902,0.182,0.372,0.207,0.0679,0.135,12.6,1.36,19.6,102,136,0,0|
0.444,0.215,0.71,0.136,0.452,0.911,0.255,0.724,0.796,2.54,16.2,7.66,96.6,154,0,0|
0.481,0.88,0.39,0.0477,0.0941,0.632,0.265,0.26,0.114,2.25,44.9,73.8,16.6,148,0,0|
0.662,0.903,0.491,0.292,0.379,0.131,0.894,0.876,0.238,23.4,56.6,82.9,17.6,92.2,0,0|
0.693,0.0252,0.737,0.72,0.179,0.922,0.422,0.356,0.803,2.19,0.851,1.08,160,72.5,0,0|
```

```
{
  "user_id": 1726817896,
  "iterador": 0,
  "referencia": [1,1,0,0,0,1,1,1,1,1],
  "solMod": [
    [
      0.331,
      0.0098,
      0.599,
```

```
<?xml version="1.0"?>
<root>
  <user_id>1726817896</user_id>
  <iterador>0</iterador>
  <referencia>
    <row>1</row>
    <row>1</row>
    <row>1</row>
    <row>1</row>
    <row>1</row>
    <row>1</row>
    <row>0</row>
    <row>0</row>
    <row>0</row>
    <row>0</row>
  </referencia>
  <solMod>
    <row>
      <column>0.463</column>
      <column>0.545</column>
      <column>0.694</column>
      <column>0.61</column>
```

Figura 11 Soluciones generadas por el framework MS-DOSS en formatos CSV, JSON y XML

En la figura 11 se muestran las soluciones generadas por el framework MS-DOSS los cuales serán recibidos por las distintas aplicaciones de la arquitectura propuesta.

En la figura 12 se encuentra el módulo de MS-DOSS el cual se encapsulo en un servidor de NodeJS para comunicar las aplicaciones. El MS-DOSS se divide en 3 microservicios, el optimizador donde se generan las nuevas soluciones aleatorias; el microservicio selector donde con ITOPSIS se ordenan las soluciones de mejor a peor, y el microservicio de desagregación de preferencias en el cual dado un conjunto de soluciones evaluadas por el DM se obtienen las preferencias de este.

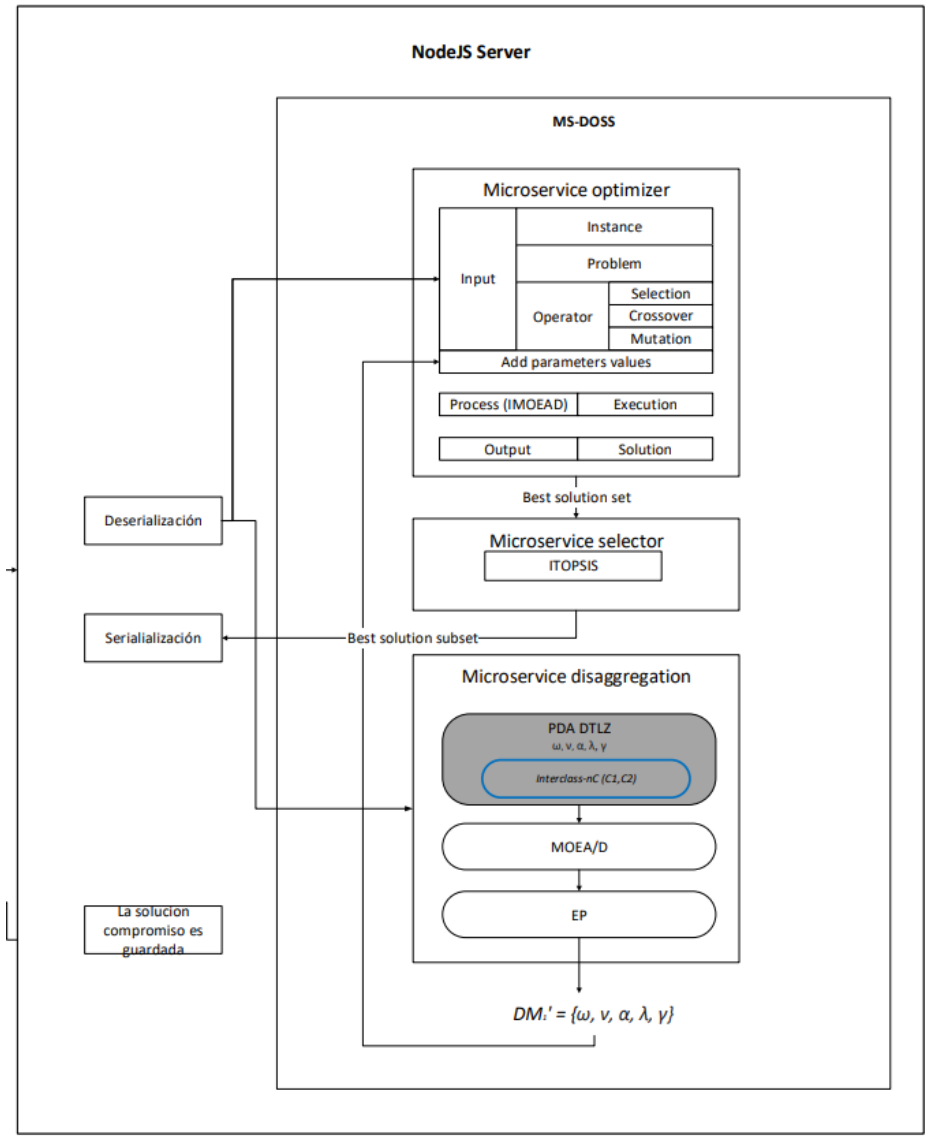


Figura 12 Arquitectura de solución (MS-DOSS)

En este servidor también se realizan las tareas de serialización y deserialización de los datos, la deserialización es necesaria para enviar los datos recibidos al framework para la generación de las nuevas soluciones tomando en cuenta las preferencias del DM, y una vez obtenidas estas soluciones esta se regresa al módulo de telegram para ser mostradas al usuario o enviarlas a VisTHAA.

En la figura 13 se muestra el módulo de VisTHAA el cual también fue encapsulado en un servidor de NodeJS para su comunicación, en este dadas las soluciones se sacan datos estadísticos como el promedio, desviación estándar y varianza, así como una gráfica para que el decisor observe como sus selecciones van afectando a las soluciones generadas.

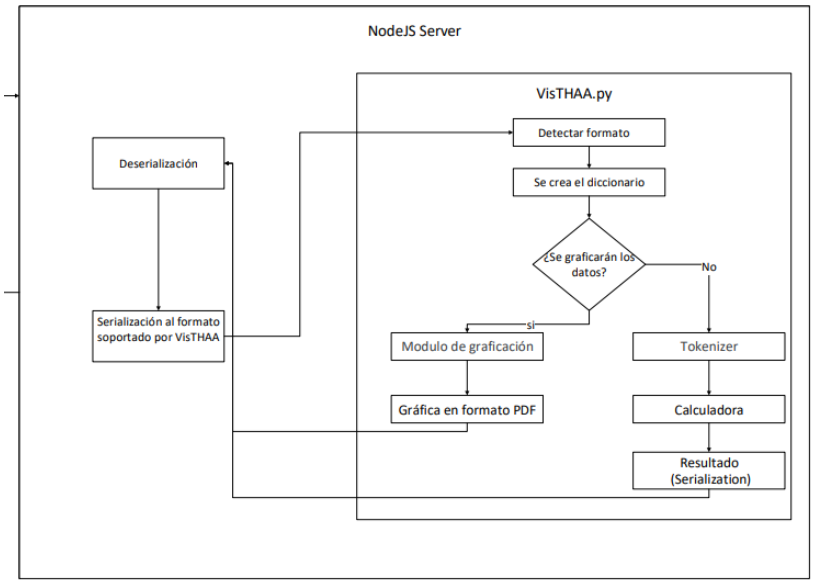


Figura 13 Arquitectura de solución (VisTHAA)

En el módulo de VisTHAA es necesaria la serialización a un formato soportado por VisTHAA, una vez en VisTHAA es generado un diccionario con los datos el cual ayuda a realizar las gráficas u operaciones estadísticas y con la ayuda de este diccionario los resultados son exportados fácilmente a cualquiera de los formatos de intercambio utilizado CSV, JSON o XML.

A continuación, se explican los pasos que se realizan en una interacción de la herramienta:

Algoritmo 1 interacción entre módulos

1. Con el comando de inicio primero se serializa el id del usuario al formato elegido y se envía al servidor donde está el framework de MS-DOSS.
2. En el servidor, se lleva a cabo la deserialización del ID del usuario y se ejecuta el microservicio optimizador que genera las soluciones aleatorias, estas se ordenan por ITOPSIS, se serializan al formato y se envían como respuesta.
3. Una vez que los datos han sido deserializados antes de enviarlos al usuario, estos se vuelven a serializar y se envían al servidor donde esta VisTHAA.
4. En este servidor se hace la deserialización y se serializan al formato soportado por VisTHAA, en el microservicio de VisTHAA se hace una detección de formato y se crea un diccionario de las soluciones, y con este se grafican los datos o se obtienen los datos estadísticos (promedio, desviación estándar, mediana y varianza), y el archivo resultante se envía como respuesta.
5. Una vez recibidas las respuestas de VisTHAA estas se muestran al usuario, seguido de las 10 primeras soluciones, de estas el usuario selecciona su favorita (la solución compromiso) la cual se envía al servidor de MS-DOSS para ser almacenada.

6. Después de esto en el servidor de Backend de telegram se serializa el id del usuario junto con el iterador (iniciado en 0) para solicitar las nuevas soluciones dando como iniciales las anteriores.
7. En el servidor de MS-DOSS se lleva a cabo la deserialización de los datos y se ejecuta el microservicio optimizador, las soluciones generadas se ordenan por ITOPSIS, se serializan al formato y se envían como respuesta.
8. Se repiten los pasos 3 y 4
9. Una vez mostrados los datos estadísticos y la gráfica se le van mostrando una a una las 10 soluciones al decisor preguntándole si le gustan o no esa solución (generando así el conjunto de referencia.)
10. Una vez hecho esto se serializan las soluciones, el conjunto de referencia y el iterador, y se envían al servidor de MS-DOSS.
11. En el servidor de MS-DOSS se lleva a cabo la deserialización de los datos y se ejecuta el microservicio de desagregación de preferencias que obtiene los valores de los pesos que representan las preferencias del decisor, en seguida se ejecuta el optimizador con estos nuevos pesos, las soluciones generadas se ordenan por ITOPSIS, se serializan al formato y se envían como respuesta (y se aumenta el iterador en uno).
12. Se repiten los pasos 3-4 seguido de los 9-11 mientras el iterador sea menor a tres.
13. Finalmente, en lugar de mostrarle las 10 mejores soluciones, se muestra la primera solución indicando al decisor que es la mejor solución para él.

Capítulo 4
Experimentación y resultados

En esta sección se presentará la información necesaria para validar el objetivo general, a través de los experimentos propuestos.

4.1 Condiciones experimentales

Para la experimentación de este trabajo se ha utilizado este equipo:

- Procesador AMD Ryzen 5 5600G with Radeon Graphics 3.90GHZ
- Ram 16.0 GB (13.9 GB utilizable)
- Sistema operativo 64 bits, procesador x64 SO Windows 11
- Javascript NodeJS versión 16.18.0
- Framework MS-DOSS
- Visthaa
- Chatbot

4.2 Diseño de experimento

Para comprobar el rendimiento de los formatos de intercambio de datos se hizo uso de la aplicación presentada en el capítulo 3, para la experimentación se realizaron tres experimentos:

1) Validar la eficiencia de los formatos de intercambio de datos a través de la comunicación entre el chatbot y framework optimización.
2) Validar la eficiencia de los formatos de intercambio de datos a través de la comunicación entre el chatbot, framework optimización y la herramienta VisTHAA.
3) Validar la eficiencia de los formatos de intercambio de datos en el proceso de serialización y deserialización de los datos.

Las métricas utilizadas fueron las siguientes:

- Peso del archivo en bytes: esta se calculó obteniendo el tamaño del archivo generado al serializar los datos al formato indicado:
- Tiempo en segundos: se mide el tiempo que se tarda en serializar o de serializar los datos al formato indicado.
- El porcentaje de uso de CPU: se midió el porcentaje de uso utilizados durante el proceso de serializar y deserializar los datos al formato, se hizo uso de la función de cpuInfo de NodeJS.
- El uso de RAM en bytes: Se midió la RAM utilizada antes y después del proceso de serialización y deserialización.

Dadas estas métricas se hicieron las mediciones para datos de tamaño chico 100, mediano 1300 y grande 3000.

Tabla 4 Diseño del experimento

Formato	CSV			JSON			XML		
Métricas	Chico 100	Mediano 1300	Grande 3000	Chico 100	Mediano 1300	Grande 3000	Chico 100	Mediano 1300	Grande 3000
Tamaño									
Tiempo									
Uso de CPU									
Uso de RAM									

4.3 Algoritmo

El algoritmo utilizado para resolver el problema DTLZ1(descrito en el marco teorico) es el MOEA/D/O con las siguientes configuraciones:

Número máximo de evaluaciones: 100000

Tamaño de la población: 100

Tamaño de los Vecinos/vecindad T:10

Operador de cruza: cruce binario simulado (SBX)

Probabilidad de cruza: 1,0

Numero de soluciones iniciales:0 (Durante la primera iteración, posteriormente serán 10)

Umbral de credibilidad: 0.67

Umbral de mayoría: 0.67

Umbral de dominancia:0.51

Mutación polinomial con probabilidad del 0.14

4.4 Instancias utilizadas

Para la experimentación se hizo uso del problema DTLZ1 de 5 objetivos y 9 variables, para resolver este problema se hizo uso del algoritmo MOEA/D/O

4.5 Experimento 1

Objetivo: Validar la eficiencia de los formatos de intercambio de datos a través de la comunicación entre el chatbot y framework optimización.

Hipótesis: Existe diferencia entre el uso de uno u otro formato de intercambio de datos en la interacción entre 2 aplicaciones.

Para este experimento se consideró el intercambio de datos entre el chatbot y el framework, se realizaron treinta corridas. Para comprobar si hay diferencias significativas en el uso de los diferentes formatos de intercambio de datos. Para validar las diferencias significativas se aplicó una prueba no paramétrica de rangos alineados de Friedman, y posteriormente se usó el posthoc de Shaffer para realizar las comparaciones entre formatos. Dando como resultados los siguientes datos:

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
100	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
100	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	Uso de RAM	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	Tiempo	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
1300	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON
1300	Uso de RAM	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Tiempo	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
3000	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	Uso de RAM	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	Tiempo	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tablas 5, 6 y 7 Resultados de la prueba no paramétrica y prueba posthoc del experimento 1

Como se observa en la tabla 5 se recopilan los resultados de los datos de tamaño chico, en la columna H0 se observa que la prueba de hipótesis se rechaza por lo que, si existen diferencias entre los grupos, para observar estas diferencias entre los grupos las siguientes columnas tienen los resultados de la prueba post hoc para comprobar las diferencias entre estos grupos.

En el caso de CSV vs JSON se observa que no hay diferencias significativas para la métrica del uso de CPU, así como en la métrica del uso de RAM, mientras que en la métrica del peso y del tiempo estas diferencias entre CSV y JSON si son significativas (los datos recopilados del promedio de las 30 corridas se encuentran en los anexos) siendo CSV mejor en el caso del peso y JSON en la métrica del tiempo.

En las siguientes 2 columnas están las comparaciones de CSV vs XML y JSON vs XML las cuales indican en todos los casos que la hipótesis se rechaza por lo que las diferencias entre estos son significativas, esto con los datos recopilados de las 30 corridas (los cuales se incluyen en los anexos) se observa que CSV y JSON tienen mejor rendimiento que XML.

En tabla 6 se encuentran los datos para los archivos de tamaño mediano, nuevamente si existen diferencias entre los grupos, y en las comparaciones entre CSV y JSON se observa que estas diferencias no son significativas para las métricas del uso de CPU y RAM, mientras que la métrica del peso JSON fue mejor que CSV con diferencias significativas y CSV para el tiempo.

De los datos recopilados XML fue el que tuvo peor rendimiento en cada una de las métricas, esto debido a que XML hace uso de etiquetas para la identificación de los atributos de los datos, por lo que las comparaciones en los datos presentados serán entre CSV y JSON.

Como se observa en las tablas de los datos recopilados, en todos los casos revisados las métricas del uso de CPU y uso de RAM no hay diferencias entre el uso de CSV o JSON, si bien se ve que JSON parece tener un mayor rendimiento que CSV esta diferencia no es significativa.

Para la métrica del peso se puede ver que CSV es el ideal para datos de tamaño pequeño, así como grande, JSON tiene buen rendimiento en datos de tamaño intermedio, pero para más grandes CSV es mejor.

Finalmente, en la métrica del tiempo se observó que, para datos de tamaño pequeño, así como grandes JSON tiene un mejor rendimiento que CSV, siendo este último adecuado para datos de

tamaño intermedio.

De esta experimentación se puede concluir que si existe diferencias entre el uso de uno u otro formato de intercambio de datos. A continuación, se destacan estas observaciones:

- XML tuvo el peor desempeño de los tres formatos revisados, en cada una de las métricas para los distintos tamaños, esto debido a que en su declaración se hace uso de etiquetas, lo cual aumenta el tiempo de procesado, así como el peso del archivo.
- Para las métricas del uso de CPU y el uso de memoria RAM no existen diferencias significativas entre el uso de JSON o CSV.
- En el caso del peso, CSV tiene mejor rendimiento que JSON con excepción de datos de tamaño intermedio.
- Para el tiempo JSON obtuvo mejor rendimiento que CSV con excepción de datos de tamaño intermedio.

4.6 Experimento 2

Objetivo: Validar la eficiencia de los formatos de intercambio de datos a través de la comunicación entre el chatbot, framework optimización y la herramienta de análisis y diagnóstico.

Hipótesis: Existe diferencia entre el uso de uno u otro formato de intercambio de datos en la interacción entre 3 aplicaciones.

Para este experimento se consideró el intercambio de datos entre el chatbot, el framework y la herramienta de análisis y diagnostico VisTHAA, se realizaron treinta corridas. Para comprobar si hay diferencias significativas en el uso de los diferentes formatos de intercambio de datos. Para validar las diferencias significativas se aplicó una prueba no paramétrica de rangos alineados de Friedman, y posteriormente se usó el posthoc de Shaffer para realizar las comparaciones entre formatos. Dando como resultados los siguientes datos:

De los datos recopilados nuevamente XML fue el que tuvo peor rendimiento en cada una de las métricas, esto debido a lo anteriormente explicado en el experimento anterior.

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
100	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	Uso de RAM	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON
100	Tiempo	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
1300	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	Uso de RAM	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	Tiempo	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV

Tamaño	Metrica	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
3000	Uso de CPU	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	Peso	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	Uso de RAM	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	Tiempo	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tabla 8, 9, 10 Resultados de la prueba no paramétrica y prueba posthoc del experimento 2

Como se revisó en el experimento anterior en esta ocasión nuevamente la prueba no paramétrica indica que, si hay diferencias entre los grupos como se observa en la columna H0, en la tabla 8 se muestran los datos recopilados de los archivos de tamaño pequeño, en la tabla 9 los de tamaño mediano y en la ultima los datos de tamaño grande.

En la tabla 8 se puede observar que para el uso de CPU JSON es mejor que CSV, pero estas diferencias no son significativas, en el caso del peso CSV es mejor que JSON con diferencias significativas y en el uso de memoria RAM JSON es mejor que CSV con diferencias significativas, por último, en la métrica del tiempo no hay diferencias entre el uso de CSV o JSON.

En la tabla 9 se observa que en el uso de CPU tiene mejores resultados, pero no son significativos, para el caso del peso nuevamente CSV es el mejor al igual que en la RAM pero esta vez con diferencias significativas, y en el tiempo también pero no con diferencias significativas.

Como se observa en las tablas de los datos recopilados, en todos los casos revisados las métricas del uso de CPU no hay diferencias entre el uso de CSV o JSON, si bien se ve que JSON parece tener un mayor rendimiento que CSV esta diferencia no es significativa.

Para la métrica del peso se puede ver que CSV es el ideal para los datos de los distintos tamaños revisados, siendo mejor que JSON y XML con diferencias significativas.

En la métrica del uso de memoria se observó que JSON tiene mejor rendimiento para datos de tamaño pequeño, mientras que CSV es mejor en los de tamaño mediano y grande.

En el caso del tiempo se observó que no hay diferencias significativas entre el uso de JSON o CSV en los datos de tamaño pequeño y mediano, mientras en los de tamaño grande JSON es mejor con diferencias significativas.

De esta experimentación se puede concluir que si existe diferencias entre el uso de los distintos formatos de intercambio de datos entre las tres aplicaciones en este caso se observó un mejor rendimiento del formato CSV. A continuación, se destacan estas observaciones:

- Nuevamente XML obtuvo el peor desempeño de los tres formatos revisados para los distintos tamaños de datos en cada una de las métricas.
- A pesar de que JSON tiene un mejor rendimiento en el uso de CPU esta diferencia no es significativa con respecto a CSV.
- En el caso de memoria RAM CSV tuvo mejor rendimiento al incorporar una nueva aplicación (VisTHAA en este caso) siendo estas diferencias significativas.
- Finalmente, para el tiempo JSON es mejor en datos de tamaño grande, y no hay diferencias significativas entre el uso de CSV y JSON en datos de tamaño pequeño y mediano.

4.7 Experimento 3

Objetivo: Validar la eficiencia de los formatos de intercambio de datos en el proceso de Serializar o de serializar los datos a través de la comunicación entre el chatbot y framework, así como en la interacción de las 3 aplicaciones.

4.7.1 ¿Por qué separar el proceso en serialización y deserialización de los datos?

Debido a que no siempre la comunicación es de manera bidireccional, los datos recolectados fueron también separados en serialización y deserialización para ver el desempeño de los formatos en estos casos.

La serialización es definida como el proceso de convertir el estado de un objeto en un formato que se pueda almacenar o transportar. El complemento de serialización es deserialización, que convierte una secuencia en un objeto. Juntos, estos procesos permiten almacenar y transferir datos (Microsoft, 2023),

Las tablas completas se encuentran en los anexos, en esta sección se muestra un resumen de estas.

Hipótesis: Existen diferencias entre el uso de uno u otro formato de intercambio de datos en la interacción entre 3 aplicaciones, en los procesos de serialización y deserialización de los datos

4.7.2 Interacción entre Chatbot, Framework y herramienta de análisis y diagnostico

En este caso se consideró el intercambio de datos entre el chatbot, el framework, se realizaron treinta corridas, para comprobar si hay diferencias significativas entre el uso de los formatos, se utilizó la prueba no paramétrica de rangos alineados de Friedman y a su vez se hizo uso del posthoc de Shaffer para realizar las comparaciones entre los formatos. Dando como resultados los siguientes datos: (Aquí nuevamente XML obtuvo los peores resultados, por lo que la siguiente tabla muestra el resumen de las comparaciones entre CSV y JSON, las tablas completas se encuentran en los anexos.)

Tamaño: 100				
Estado de los datos	Uso de CPU	Peso	Uso de RAM	Tiempo
Serialización	CSV	CSV	JSON	JSON
Deserialización	CSV	CSV	JSON	CSV

Tamaño: 1300				
Estado de los datos	Uso de CPU	Peso	Uso de RAM	Tiempo
Serialización	JSON	CSV	JSON	JSON
Deserialización	CSV	CSV	JSON	CSV

Tamaño: 3000				
Estado de los datos	Uso de CPU	Peso	Uso de RAM	Tiempo
Serialización	JSON	CSV	JSON	JSON
Deserialización	CSV	CSV	CSV	CSV

Tablas 11, 12, 13 Resultados de la interacción de 2 aplicaciones en el experimento 3, separados en serialización y deserialización

En tabla 11 se tiene los datos de los archivos de tamaño pequeño, esta se divide en el estado de los datos serialización y deserialización como se observa no hay diferencias entre el uso de CSV o JSON para el uso de CPU, en el caso del peso CSV fue el mejor en los 2 casos revisados, en el caso del uso de RAM JSON fue el mejor pero en el caso de la deserialización esta diferencia no es significativa, y finalmente para el tiempo JSON es el mejor en el caso de la serialización con diferencias significativas y CSV para la deserialización pero este último no hay diferencias significativas respecto a JSON.

En la tabla 12 se encuentran los datos de tamaño mediano, en los que se observa que para el caso del uso de CPU JSON presento mejores resultados que CSV, pero estos no son significativos, y en el caso de la deserialización CSV es mejor pero igualmente no son significativos; para el peso CSV fue el mejor en Serialización como en deserialización; en el uso de RAM JSON obtuvo los mejores resultados, pero no son significativos respecto a CSV; y en el caso del tiempo JSON es mejor en el caso de serialización y CSV en deserialización ambos con diferencias significativas.

Por último, en la tabla 13 se encuentran los datos para tamaño pequeño, como se observa respecto a la tabla anterior se presentó el mismo caso para el uso de CPU, en el peso nuevamente CSV es mejor en los dos casos para la métrica del peso, en el uso de RAM esta vez CSV fue mejor que JSON para la deserialización, pero sin diferencias significativas, y en el tiempo otra vez JSON es mejor en el caso de la serialización y CSV para la deserialización.

Como se observa en las tablas, en la métrica del uso de CPU no existen diferencias significativas entre el uso de CSV o JSON.

Para la métrica del peso CSV Fue el mejor tanto en la serialización como la deserialización para cada uno de los tamaños de datos revisados.

En el caso del uso de memoria RAM, JSON es el mejor en el caso de serialización para datos de tamaño pequeño, en los siguientes casos no existe diferencias significativas entre su uso o el de CSV.

Y por último en la métrica del tiempo JSON es el mejor para los casos de serialización, y CSV en la deserialización a excepción de la deserialización de datos de tamaño 100 en la cual no hay diferencias significativas.

4.7.3 Interacción entre Chatbot y Framework

En este caso se consideró el intercambio de datos entre el chatbot y el framework, se realizaron treinta corridas, para comprobar si hay diferencias significativas entre el uso de los formatos, se utilizó la prueba no paramétrica de rangos alineados de Friedman y a su vez se hizo uso del posthoc de Shaffer para realizar las comparaciones entre los formatos. Dando como resultados los siguientes datos:

Tamaño 100				
Estado de los datos	**CPU**	**PESO**	**RAM**	**TIEMPO**
Serialization	JSON	CSV	JSON	JSON
Deserialization	JSON	CSV	JSON	JSON

Tamaño 1300				
Estado de los datos	**CPU**	**PESO**	**RAM**	**TIEMPO**
Serialization	JSON	CSV	CSV	JSON
Deserialization	JSON	CSV	CSV	CSV

Tamaño 3000				
Estado de los datos	**CPU**	**PESO**	**RAM**	**TIEMPO**
Serialization	JSON	CSV	CSV	JSON
Deserialization	JSON	CSV	CSV	JSON

Tablas 14, 15, 16 Resultados de la interacción de 3 aplicaciones en el experimento 3, separados en serialización y deserialización

Como se observa en las tablas, en la métrica del uso de CPU no existen diferencias significativas entre el uso de CSV o JSON a pesar de que JSON tenga mejor rendimiento.

Para la métrica del peso, CSV es el mejor en todos los casos revisados con diferencias significativas.

En el caso del uso de memoria RAM para la serialización de los datos JSON es el mejor en datos de tamaño pequeño, pero para la deserialización de los datos no tiene diferencias significativas con respecto a CSV. En datos de tamaño mediano y grande CSV es el mejor pero únicamente siendo esta diferencia significativa para la deserialización de los datos medianos.

Para el caso del tiempo JSON tiene mejor rendimiento en datos de tamaño pequeño, pero no tiene diferencias significativas, en datos de tamaño mediano en el caso de serialización JSON es mejor sin diferencias significativas, y para la deserialización CSV es mejor con diferencias significativas, en datos de tamaño grande JSON es el mejor pero solo en la serialización esta diferencia es significativa.

Capítulo 5
Conclusiones y trabajos futuros

En la presente investigación se diseñó una arquitectura de integración entre aplicaciones de optimización haciendo uso de microservicios. Para lograr el diseño de la arquitectura se logró el objetivo general que es analizar el impacto de distintos formatos para el intercambio de datos aplicados a la interacción entre un Chat-Bot, un optimizador y una herramienta de visualización y diagnostico para propósito de comunicación de información.

Además, también se cumplieron cada uno de los objetivos especificos los cuales se lograron de la siguiente forma:

- **Caracterización de la información que se va a transferir**: como se observó en la propuesta de solución se observaron los datos que iban a ser transmitidos entre las aplicaciones y se definió la estructura de los datos en los distintos formatos CSV, JSON y XML.
- **Analizar formatos de intercambio**: se hizo el análisis de los formatos de intercambio de datos como se observó en el marco teorico, se optó por el uso de los formatos CSV, JSON y XML debido a que son los más populares además de que otros formatos revisados son variaciones de estos.
- **Establecer métricas de calidad para el desempeño de los formatos de intercambio de datos en el contexto**: Las métricas de calidad utilizadas fueron utilizadas de las más comunes en los trabajos del estado del arte tomando en cuenta que en este trabajo los datos en lugar de ser almacenados en una base de datos serian procesados por distintas aplicaciones, se hizo énfasis en hacer uso de las métricas únicamente en factores que afecten el uso de un formato u otro, por ejemplo el tiempo que se tarda en serializar los datos al formato CSV, JSON o XML, ya que hacer la medición desde la petición de la solicitud de los datos implica medir el tiempo en que tardan en llegar los datos debido a la base de datos utilizada la velocidad de internet u otros factores externos.
- **Analizar y validar el desempeño que los formatos de intercambio en función de las métricas definidas**: en la sección de experimentación y resultados se validó el desempeño de los formatos para las distintas métricas en los tamaños de datos definidos.

Con los datos recopilados se cumple la hipótesis propuesta:

H0: JSON no es el mejor formato de intercambio de datos entre un Chat-Bot, un optimizador y una herramienta de visualización y diagnóstico para propósito de comunicación de información específica.

Como se observó CSV presento mejores resultados que JSON en la métrica del peso de archivo con

diferencias significativas en todos los casos revisados en esta investigación.

Así también como para la RAM y tiempo de la deserialización de los datos de tamaño mediano, respecto a las métricas del porcentaje de uso de CPU y el uso de memoria RAM en la mayoría de los casos no hay diferencias significativas entre el uso de JSON o CSV.

Contribuciones de este proyecto:

- Arquitectura de integración entre un chatbot, un framework de optimización y una herramienta de análisis y diagnostico haciendo uso de distintos formatos de intercambio de datos CSV, JSON y XML.
- Análisis del uso de los formatos de intercambio de datos CSV, JSON Y XML para la interacción entre dos aplicaciones de optimización.
- Análisis del uso de los formatos de intercambio de datos CSV, JSON Y XML para la interacción entre tres aplicaciones de optimización.
- Análisis del uso de los formatos de intercambio de datos CSV, JSON Y XML para la interacción entre dos o tres aplicaciones de optimización, en la serialización y deserialización de los datos.

Trabajos Futuros:

- Análisis del uso de un conjunto de distintos formatos de intercambio de datos, por ejemplo: Uso de CSV entre dos aplicaciones y el uso de JSON entre la primera y una tercera aplicación.
- Integración de más aplicaciones a la arquitectura propuesta.
- Comparación de datos serializados CSV, JSON, XML con respecto a otros formatos de serialización (Binaria).
- Incorporación de nuevos módulos de VisTHAA.

Participación en eventos.

Seminario de cómputo Inteligente en el Instituto Tecnológico de Ciudad Madero 2022, 2023.

1er Coloquio Interinstitucional de Investigación Básica y de Frontera, en la Universidad Autónoma de Tamaulipas

The Technichal program of the International Seminar of Computacional Intelligence ISCI 2023. With the Talk: Analysis of the efficiency of data Exchange formats on decision-aid tools.

Encuentro de jóvenes investigadores Tamaulipas 2023

International Workshop on Artificial Intelligence and Analytics - Eureka 2023

Bibliografía

Anderson, D. R., Sweeney, D. J., & Williams, T. A. (2019). *Estadística para administración y economía.* Cengage Learning Editores.

Balderas, F. F. (2016). International Journal of combinatorial optimization problems and informatics. *7*(3), 101-118.

Breje, A.-R., Győrödi, R., Győrödi, C., Doina, Z., & Pecherle, G. (2018). *Comparative study of data sending methods for XML and JSON models. International Journal of Advanced Computer Science and Applications.* Oradea, Rumania. doi:10.14569/IJACSA.2018.091229

Doumpos, M., & Zopounidis, C. (2004). *Developing sorting models using preference disaggregation.* Elsevier.

Fernández, E., Rangel Valdez, N., Cruz Reyes, L., Gomez Santillan, C. G., & Coello Coello, C. (2022). Preference incorporation in MOEA/D using an outranking approach with imprecise model parameters. *Swarm and Evolutionary Computation*, 1-24. doi:https://doi.org/10.1016/j.swevo.2022.101097.

Hernández Vicente, P. E. (2023). *Arquitectura de interacción entre decisores y framework de optimización.* Ciudad Madero.

JSON. (s.f.). Obtenido de JSON: http://www.json.org/json-es.html

Liu, J., & Xu, Y. (2022). T-Friedman Test: A New Statistical Test for Multiple Comparison with an Adjustable Conservativeness Measure. *International Journal of Computational Intelligence Systems, 15*(1), 29. doi:10.1007/s44196-022-00083-8

Microsoft. (15 de Febrero de 2023). *Microsoft NET*. Obtenido de Learn Microsoft: https://learn.microsoft.com/es-es/dotnet/standard/serialization/

O'Reilly Media. (10 de Febrero de 1998). *XML*. Obtenido de https://www.xml.com/pub/a/98/10/guide0.html

Ponce Nájera, J. G. (2021). *Análisis de desempeño usando visthaa aplicado a un algoritmo metaheurístico de un framework de optimización.* Ciudad Madero.

Repici, D. (18 de Octubre de 2002). Obtenido de CSV Comma Separated Value File Format: https://www.creativyst.com/Doc/Articles/CSV/CSV01.shtml

Riantoa, M. A., Gunawana, R., Darmawanb, I., & Rahmatulloha, A. (2021). *Comparison of JSON and XML Data Formats in Document Stored.* Tasikmalaya, Indonesia. doi:10.18517/ijaseit.11.3.11570

Šabanović, M., Saračević, M., & Azizović, E. (2016). *Comparative analysis of AMF, JSON and XML technologies.* Mostar. doi:10.21533/pen.v4i2.57

Sánchez de la Paz, L. N. (2016). *Desarrollo de un protocolo de comunicación para un framework de apoyo a la toma de decisiones.* Ciudad Madero.

Wiedmaier, B. (2017). Post Hoc Tests. En S. Publications, *The SAGE Encyclopedia of Communication Research Methods.* SAGE Publications. doi:https://doi.org/10.4135/9781483381411

ANEXOS

Datos recopilados de las 30 corridas del experimento 1 interacción entre chatbot y framework MS-DOSS

CSV			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	4.48E-04	2.38E-03	4.92E-03
Uso de CPU (%)	1.09686E-07	6.11725E-07	7.51523E-07
Uso de RAM (B)	59272.53333	152354.1333	186794.6667
Peso (B)	4422.394444	54043.64167	123829.3444

JSON			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	3.44E-04	1.89E-03	3.92E-03
Uso de CPU (%)	4.16908E-07	5.23372E-07	5.06657E-07
Uso de RAM (B)	35328	113413.6889	123204.2667
Peso (B)	4476.775	55090.69444	126230.3222

XML			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	4.12E-03	3.00E+00	3.00E+00
Uso de CPU (%)	1.05288E-05	1.84486E-05	2.14756E-05
Uso de RAM (B)	200527.6444	7640695.467	13262108.44
Peso (B)	23508.46111	284147.8667	654386.0917

Tamaño	100
Metrica	Mejor Formato
Tiempo	JSON
Uso de CPU	CSV
Uso de Ram	JSON
Peso	CSV

Tamaño	1300
Metrica	Mejor Formato
Tiempo	CSV
Uso de CPU	JSON
Uso de Ram	JSON
Peso	CSV

Tamaño	3000
Metrica	Mejor Formato
TIEMPO	JSON
USOCPU	JSON
USORAM	JSON
PESOBUFFER	CSV

Estos datos corresponden al promedio de las 30 corridas realizadas.

Datos recopilados de las 30 corridas del experimento 2 interacción entre chatbot, framework MS-DOSS y la herramienta de análisis y diagnostico VisTHAA.

CSV			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	3.05E-01	2.81E-01	3.10E-01
Uso de CPU (%)	0.003950831	0.00868107	0.031662216
Uso de RAM (B)	200716.253	301710.6598	793291.9248
Peso (B)	4944.178205	61964.23932	143516.7786

Tamaño	100
Metrica	Mejor Formato
Tiempo	JSON
Uso de CPU	JSON
Uso de RAM	JSON
Peso	CSV

JSON			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	3.01E-01	2.87E-01	2.85E-01
Uso de CPU (%)	0.001224222	0.006677833	0.019865461
Uso de RAM (B)	103185.9419	415624.9709	1097265.887
Peso (B)	5139.361111	63204.08034	145733.4765

Tamaño	1300
Metrica	Mejor Formato
Tiempo	CSV
Uso de CPU	JSON
Uso de RAM	CSV
Peso	CSV

XML			
Metrica	**100**	**1300**	**3000**
Tiempo (s)	4.26E-01	1.50E+00	2.91E+00
Uso de CPU (%)	0.282731551	3.532223813	7.448565349
Uso de RAM (B)	2254793.737	31806355.47	65644782.06
Peso (B)	50536.8812	705162.9744	1495777.785

Tamaño	3000
Metrica	Mejor Formato
Tiempo	JSON
Uso de CPU	JSON
Uso de RAM	CSV
Peso	CSV

Estos datos corresponden al promedio de las 30 corridas realizadas.

Datos recopilados de las 30 corridas del experimento 3 interacción entre chatbot, framework MS-DOSS y la herramienta de análisis y diagnóstico, separados por serialización y deserialización.

Serialización			
CSV			
Metrica	100	1300	3000
Tiempo (s)	3.37E-04	2.49E-03	5.22E-03
Uso de CPU (%)	0.00	9.51E-07	9.43E-07
Uso de RAM (B)	51382.04	165717.33	253758.58
Peso (B)	4422.39	54043.64	123829.34

Deserialización			
CSV			
Metrica	100	1300	3000
Tiempo (s)	5.59E-04	2.27E-03	4.61E-03
Uso de CPU (%)	2.19E-07	2.72E-07	5.60E-07
Uso de RAM (B)	67163.02	138990.93	119830.76
Peso (B)	4422.39	54043.64	123829.34

Serialización			
JSON			
Metrica	100	1300	3000
Tiempo (s)	1.09E-04	1.21E-03	2.78E-03
Uso de CPU (%)	4.48E-07	3.85E-07	4.14E-07
Uso de RAM (B)	11730.49	91272.53	87142.40
Peso (B)	4476.78	55090.69	126230.32

Deserialización			
JSON			
Metrica	100	1300	3000
Tiempo (s)	5.79E-04	2.57E-03	5.07E-03
Uso de CPU (%)	3.86E-07	6.62E-07	5.99E-07
Uso de RAM (B)	58925.51	135554.84	159266.13
Peso (B)	4476.78	55090.69	126230.32

Serialización			
XML			
Metrica	100	1300	3000
Tiempo (s)	4.23E-03	3.44E-02	7.00E-02
Uso de CPU (%)	1.09E-05	1.51E-05	1.69E-05
Uso de RAM (B)	252154.31	13940553.96	25319913.24
Peso (B)	23508.46	284147.87	654386.09

Deserialización			
XML			
Metrica	100	1300	3000
Tiempo (s)	4.01E-03	3.36E-02	6.92E-02
Uso de CPU (%)	1.01E-05	2.18E-05	2.61E-05
Uso de RAM (B)	148900.98	1340836.98	1204303.64
Peso (B)	23508.46	284147.87	654386.09

Tamaño 100 Serialización	
Metrica	Mejor Formato
TIEMPO	JSON
USOCPU	CSV
USORAM	JSON
PESO	CSV

Tamaño 100 Deserialización	
Metrica	Mejor Formato
TIEMPO	CSV
USOCPU	CSV
USORAM	JSON
PESO	CSV

Tamaño 1300 Serialización	
Metrica	Mejor Formato
TIEMPO	JSON
USOCPU	JSON
USORAM	JSON
PESO	CSV

Tamaño 1300 Deserialización	
Metrica	Mejor Formato
TIEMPO	CSV
USOCPU	CSV
USORAM	JSON
PESO	CSV

Tamaño 3000 Serialización	
Metrica	Mejor Formato
TIEMPO	JSON
USOCPU	JSON
USORAM	JSON
PESOBUFI	CSV

Tamaño 3000 Deserialización	
Metrica	Mejor Formato
TIEMPO	CSV
USOCPU	CSV
USORAM	CSV
PESOBUFI	CSV

Estos datos corresponden al promedio de las 30 corridas realizadas.

Resultados de la prueba estadística de rangos alineados de Friedman con la prueba posthoc de Shaffer.

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
100	Uso de CPU	Serialización	0.00024	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
100	Peso	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	Uso de RAM	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON
100	Tiempo	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
100	Uso de CPU	Deserialización	0.00885	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
100	Peso	Deserialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	Uso de RAM	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	Tiempo	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
1300	Uso de CPU	Serialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Peso	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	Uso de RAM	Serialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Tiempo	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
3000	Uso de CPU	Serialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	Peso	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	Uso de RAM	Serialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	Tiempo	Serialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
1300	Uso de CPU	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
1300	Peso	Deserialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	Uso de RAM	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	Tiempo	Deserialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV

Tamaño	Metrica	Proceso	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Mejor formato
3000	Uso de CPU	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
3000	Peso	Deserialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	Uso de RAM	Deserialización	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
3000	Tiempo	Deserialización	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV

Datos recopilados de las 30 corridas del experimento 3 interacción entre chatbot y el framework MS-DOSS separados por serialización y deserialización

Tamaño 100 Serialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.002087934	0.00076524	0.21417991
Uso de CPU	0.007337141	0.00227348	0.52507081
Uso de RAM	257059.7587	84585.6508	2608787.91
Peso	2558.942063	2629.96032	13061.2032

Serialización	
Tamaño	100
Metrica	Mejor Formato
TIEMPO	json
USOCPU	json
USORAM	json
PESO	csv

Tamaño 100 Deserialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.658037116	0.651333208	0.672797463
Uso de CPU	1.35335E-07	8.3471E-08	2.42148E-06
Uso de RAM	134982.163	124886.2815	1841800.533
Peso	7726.953704	8066.99537	94258.50556

Deserialización	
Tamaño	100
Metrica	Mejor Formato
TIEMPO	json
USOCPU	json
USORAM	json
PESO	csv

Tamaño 1300 Serialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.00512314	0.00400736	1.2426608
Uso de CPU	0.01612113	0.01240085	6.55981699
Uso de RAM	194332.444	310860.394	41252594.2
Peso	30593.3294	31157.9817	267793.511

Serialización	
Tamaño	1300
Metrica	Mejor Formato
TIEMPO	json
USOCPU	json
USORAM	csv
PESO	csv

Tamaño 1300 Deserialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.6033404	0.61718765	1.79547361
Uso de CPU	9.9764E-07	9.7417E-07	3.1771E-05
Uso de RAM	426985.244	537850.311	20785743.6
Peso	98563.6343	100591.195	1215427.35

Deserialización	
Tamaño	1300
Metrica	Mejor Formato
TIEMPO	csv
USOCPU	json
USORAM	csv
PESO	csv

Tamaño 3000 Serialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.00959604	0.00814155	2.54721651
Uso de CPU	0.05880061	0.03689244	13.832987
Uso de RAM	460354.641	642369.829	83352553.2
Peso	70814.8381	71756.246	373574.448

Serialización	
Tamaño	3000
Metrica	Mejor Formato
TIEMPO	json
USOCPU	json
USORAM	csv
PESO	csv

Tamaño 3000 Deserialización			
Formatos	CSV	JSON	XML
Métricas	Chico	Chico	Chico
Tiempo	0.6600044	0.60839684	3.32479184
Uso de CPU	7.5105E-07	6.5486E-07	7.3465E-05
Uso de RAM	1181718.76	1627977.96	44985715.7
Peso	228335.709	232040.245	2805015.01

Deserialización	
Tamaño	3000
Metrica	Mejor Formato
TIEMPO	json
USOCPU	json
USORAM	csv
PESO	csv

Estos datos corresponden al promedio de las 30 corridas realizadas.

Resultados de la prueba estadística de rangos alineados de Friedman con la prueba posthoc de Shaffer.

Size	Attribute	Serialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
100	CPU	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	PESO	Serialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	RAM	Serialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON
100	TIEMPO	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON

Size	Attribute	Deserialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
100	CPU	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	PESO	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
100	RAM	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
100	TIEMPO	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON

Size	Attribute	Serialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
1300	CPU	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	PESO	Serialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	RAM	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
1300	TIEMPO	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON

Size	Attribute	Deserialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
1300	CPU	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
1300	PESO	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	RAM	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
1300	TIEMPO	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV

Size	Attribute	Serialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
3000	CPU	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	PESO	Serialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	RAM	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
3000	TIEMPO	Serialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON

Size	Attribute	Deserialization	P-Value	H0	CSV vs JSON	CSV vs XML	JSON vs XML	Best Format
3000	CPU	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	JSON
3000	PESO	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	CSV
3000	RAM	Deserialization	0.00000	H0 is rejected	H0 is accepted	H0 is rejected	H0 is rejected	CSV
3000	TIEMPO	Deserialization	0.00000	H0 is rejected	H0 is rejected	H0 is rejected	H0 is rejected	JSON