UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

BIG DATA ANALYTICS APLICADA EN LA INTEGRACIÓN DE DATOS DE INTERNET DE LAS COSAS, CASO DE USO: AGRICULTURA DE PRECISIÓN

VARGAS NEIRA ALVARO RONNY
INGENIERO DE SISTEMAS

MACHALA
2021

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

BIG DATA ANALYTICS APLICADA EN LA INTEGRACIÓN DE DATOS DE INTERNET DE LAS COSAS, CASO DE USO: AGRICULTURA DE PRECISIÓN

VARGAS NEIRA ALVARO RONNY
INGENIERO DE SISTEMAS

MACHALA
2021

UTMACH

FACULTAD DE INGENIERÍA CIVIL

CARRERA DE INGENIERÍA DE SISTEMAS

TRABAJO TITULACIÓN
PROPUESTAS TECNOLÓGICAS

BIG DATA ANALYTICS APLICADA EN LA INTEGRACIÓN DE DATOS DE INTERNET DE LAS COSAS, CASO DE USO: AGRICULTURA DE PRECISIÓN

VARGAS NEIRA ALVARO RONNY
INGENIERO DE SISTEMAS

MAZÓN OLIVO BERTHA EUGENIA

MACHALA, 26 DE ABRIL DE 2021

MACHALA
2021

BIG DATA ANALYTICS APLICADA EN LA INTEGRACIÓN DE DATOS DE INTERNET DE LAS COSAS, CASO DE USO: AGRICULTURA DE PRECISIÓN

INFORME DE ORIGINALIDAD

4%	4%	0%	1%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	vsip.info Fuente de Internet	1%
2	Submitted to Universidad Internacional de la Rioja Trabajo del estudiante	<1%
3	Submitted to Universidad Técnica de Machala Trabajo del estudiante	<1%
4	oa.upm.es Fuente de Internet	<1%
5	www.dspace.espol.edu.ec Fuente de Internet	<1%
6	docs.google.com Fuente de Internet	<1%
7	qdoc.tips Fuente de Internet	<1%
8	mafiadoc.com Fuente de Internet	<1%

CLÁUSULA DE CESIÓN DE DERECHO DE PUBLICACIÓN EN EL REPOSITORIO DIGITAL INSTITUCIONAL

El que suscribe, VARGAS NEIRA ALVARO RONNY, en calidad de autor del siguiente trabajo escrito titulado BIG DATA ANALYTICS APLICADA EN LA INTEGRACIÓN DE DATOS DE INTERNET DE LAS COSAS, CASO DE USO: AGRICULTURA DE PRECISIÓN, otorga a la Universidad Técnica de Machala, de forma gratuita y no exclusiva, los derechos de reproducción, distribución y comunicación pública de la obra, que constituye un trabajo de autoría propia, sobre la cual tiene potestad para otorgar los derechos contenidos en esta licencia.

El autor declara que el contenido que se publicará es de carácter académico y se enmarca en las dispociones definidas por la Universidad Técnica de Machala.

Se autoriza a transformar la obra, únicamente cuando sea necesario, y a realizar las adaptaciones pertinentes para permitir su preservación, distribución y publicación en el Repositorio Digital Institucional de la Universidad Técnica de Machala.

El autor como garante de la autoría de la obra y en relación a la misma, declara que la universidad se encuentra libre de todo tipo de responsabilidad sobre el contenido de la obra y que asume la responsabilidad frente a cualquier reclamo o demanda por parte de terceros de manera exclusiva.

Aceptando esta licencia, se cede a la Universidad Técnica de Machala el derecho exclusivo de archivar, reproducir, convertir, comunicar y/o distribuir la obra mundialmente en formato electrónico y digital a través de su Repositorio Digital Institucional, siempre y cuando no se lo haga para obtener beneficio económico.

Machala, 26 de abril de 2021

VARGAS NEIRA ALVARO RONNY
0704426675

DEDICATORIA

El presente proyecto está dedicado principalmente a mis padres, cuyo esfuerzo se ve reflejado en la persona que soy actualmente, todos los logros que he cosechado y éste, el más importante hasta la fecha, es producto de la motivación, valores y enseñanza, que en conjunto con algunas libertades me han permitido cumplir muchas de mis metas propuestas.

A mi hermana, demás familiares, amigos y personas que estuvieron apoyándome continuamente y trabajaron junto a mi para cumplir cada uno de los objetivos requeridos y llegar a esta etapa en mi vida.

Alvaro Ronny Vargas Neira

AGRADECIMIENTO

Agradezco a mis padres por el apoyo, paciencia y motivación durante estos años, de igual manera a mis familiares, amigos, docentes y personas que estuvieron en mi vida motivándome a seguir adelante, especialmente a la Ing. Sist. Bertha Mazón que, con su ayuda e instrucciones en los últimos años de mi carrera universitaria, me ha transmitido muchos de sus conocimientos y me ha brindado la oportunidad de trabajar a su lado para culminar este proyecto y etapa tan importante en mi vida.

A la Universidad Técnica de Machala, Facultad de Ingeniería Civil, Carrera de Ingeniería de Sistemas por aceptarme en su establecimiento y ayudarme en mi desarrollo profesional.

Alvaro Ronny Vargas Neira

RESUMEN

Cada vez más aumenta la necesidad de una mayor producción agrícola con excelentes parámetros de calidad, algo que ha supuesto un problema adicional para los agricultores provocando que apliquen el concepto de agricultura de precisión (AP), la cual consiste en brindar un mayor soporte para la gestión de cultivos gracias al análisis de los datos recolectados a través de tecnologías y medios electrónicos. Razón por la cual se plantea este proyecto que consiste en analizar grandes cantidades de datos pertenecientes a lotes de cultivos, aplicando procesos de recolección, transformación e integración de datos. Toda la arquitectura planteada para el desarrollo del prototipo está basada en cada una de las fases (pasos) de la metodología MAMBO, la cual fue propuesta por un equipo de consultores estratégicos especializados en el campo de la analítica y que vieron la necesidad de implementar mejoras en la metodología CRISP-DM para proyectos de Big Data Analytics. Se inicia con un análisis general de los sistemas de agricultura de precisión y de los datos requeridos para ser analizados a futuro, luego para el proceso de adquisición de datos, y considerando la falta de acceso a datos históricos reales, se utiliza un simulador de red de sensores inalámbricos que genere datos sintéticos como mensajes de dos tipos, configuración y lecturas, el primero, para información relacionada a cultivos, dispositivos, empresas, entre otros, y el segundo, para valores que generan los sensores. También se utiliza un middleware IoT con el objetivo de transferir los mensajes a través de protocolos de IoT, MQTT y Apache Kafka, y almacenarlos en bases de datos relacional y no relacional. Para el análisis de datos no es posible utilizar solamente el conjunto de datos generados por los sensores ya que estos simplemente contienen identificadores y los valores de cada sensor, para ello, es necesario relacionarlos con la información de las configuraciones que se encuentran almacenados en la base de datos relacional, a este proceso se lo denomina integración de datos, para ello se utiliza la herramienta Denodo, la cual permite crear un solo punto de acceso a partir de distintas fuentes de datos, el problema es que los datos que se integran no se alojan en un espacio físico y además, su volumen incrementa considerablemente al punto de tener un Big Data, es por eso que es necesario un clúster Apache

Hadoop como una infraestructura Big Data para el almacenamiento y procesamiento de grandes cantidades de datos a través de sus servicios HDFS y YARN. La aplicación de estadística básica, descriptiva y predictiva se realiza al conjunto de datos almacenados en el clúster por medio de los lenguajes de programación Python y R, un mayor enfoque está sobre los modelos de predicción en series temporales pertenecientes a la estadística predictiva, y por ello, en la evaluación del prototipo, se establecen dos escenarios, el primero consiste en la medición de tiempos de transferencia para la carga de datos en streaming, es decir, los mensajes desde que son generados por el simulador hasta su almacenamiento en HDFS, y por otro lado, el segundo escenario que consiste en evaluar los modelos de predicción aplicando métricas de calidad. Los resultados del primer escenario indican que, para grandes volúmenes de datos, los tiempos de integración son altos en comparación a los tiempos de transferencia de mensajes individuales, esto, considerando los recursos de la máquina en donde se ejecuta toda la arquitectura del prototipo, mientras que, para el segundo escenario, mucho de los modelos de predicción evaluados, presentaron resultados similares uno de otros, esto quiere decir que los modelos, en su mayoría, se ajustaron bien al conjunto de datos reales para realizar predicciones significativas de cara al proceso de toma de decisiones dentro de un proceso agrícola.

Palabras clave: Big Data Analytics, integración de datos, Internet de las cosas, agricultura de precisión, modelos de predicción, series temporales.

ABSTRACT

The need for greater agricultural production with excellent quality parameters is increasing, something that has been an additional problem for farmers, causing them to apply the concept of precision agriculture (PA), which consists of providing greater support for crop management thanks to the analysis of data collected through technologies and electronic media. This is the reason for this project, which consists of analyzing large amounts of data pertaining to crop lots, applying data collection, transformation and integration processes. The entire architecture proposed for the development of the prototype is based on each of the phases (steps) of the MAMBO methodology, which was proposed by a team of strategic consultants specialized in the field of analytics and who saw the need to implement improvements in the CRISP-DM methodology for Big Data Analytics projects. It starts with a general analysis of precision agriculture systems and the data required to be analyzed in the future, then for the data acquisition process, and considering the lack of access to real historical data, a wireless sensor network simulator is used to generate synthetic data as messages of two types, configuration and readings, the first, for information related to crops, devices, companies, among others, and the second, for values generated by the sensors. An IoT middleware is also used to transfer messages through IoT protocols, MQTT and Apache Kafka, and store them in relational and non-relational databases. For data analysis it is not possible to use only the set of data generated by the sensors as these simply contain identifiers and values of each sensor, for this, it is necessary to relate them with the information of the configurations that are stored in the relational database, this process is called data integration, for this the Denodo tool is used, The problem is that the data that is integrated is not housed in a physical space and also, its volume increases considerably to the point of having a Big Data, that is why an Apache Hadoop cluster is necessary as a Big Data infrastructure for the storage and processing of large amounts of data through its HDFS and YARN services. The application of basic, descriptive and predictive statistics is performed to the dataset stored in the cluster by means of Python and R programming languages, a major focus is on prediction models in time series belonging to predictive statistics, and therefore, in the evaluation of the prototype, Two scenarios are established, the

first one consists of measuring transfer times for loading streaming data, i.e., messages from the time they are generated by the simulator to their storage in HDFS, and on the other hand, the second scenario consists of evaluating the prediction models by applying quality metrics. The results of the first scenario indicate that, for large volumes of data, the integration times are high compared to the transfer times of individual messages, this, considering the resources of the machine where the entire architecture of the prototype is executed, while, for the second scenario, many of the prediction models evaluated, presented similar results to each other, this means that the models, for the most part, adjusted well to the real data set to make significant predictions for the decision making process within an agricultural process.

Keywords: Big Data Analytics, data integration, Internet of Things, precision agriculture, predictive modeling, time series.

CONTENIDO

Pág.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

ÍNDICE DE ALGORITMOS

Pág.

GLOSARIO

Clúster: Sistema distribuido de computadoras, también conocido como granja de computadoras, interconectadas entre sí por medio de una red comportándose como si fuese un único servidor.

Apache Hadoop: Entorno de trabajo de código abierto que permite el almacenamiento y procesamiento distribuido de grandes cantidades de datos por medio de la ejecución de aplicaciones en una red de nodos.

HDFS: Sistemas de fichero de Hadoop que permite almacenar masivas cantidades de datos estructurados, semi-estructurados y no estructurados garantizando disponibilidad y tolerancia a fallos.

YARN: Framework que permite a Hadoop el soporte de varios motores de ejecución y brinda un planificador de las aplicaciones ejecutadas en el clúster.

SSH: Protocolo de administración remota que permite el control y modificación de servidores remotos por medio de internet a través de un mecanismo de autenticación.

Big Data: Conjunto de datos en donde el tamaño, complejidad y velocidad de crecimiento perjudica la gestión, procesamiento y análisis por medio de tecnologías o herramientas.

Big Data Analytics: Es el descubrimiento de tendencias o patrones en grandes cantidades de datos mediante el uso de software.

Apache Spark: Framework de código abierto que consta de distintas APIs para el procesamiento de datos distribuidos.

IoT: Internet de las cosas es la agrupación de dispositivos que se interconectan por medio de una red con la finalidad de que cualquier dispositivo pueda interactuar con otros.

WSN: Red de sensores inalámbricos, es un grupo de sensores que forman parte de una red que monitorean y envían información sobre condiciones físicas y ambientes.

MQTT: Protocolo de comunicación máquina a máquina en el que cada conexión se mantiene abierta y es reutilizada cada vez que se requiera, a diferencia de una petición HTTP en donde cada transferencia es mediante una conexión.

Apache Kafka: Plataforma distribuida de transferencia de datos para publicar, almacenar y procesar datos en tiempo real, permite la gestión de flujos de datos de distintas fuentes y distribuirlo a varios usuarios.

Agricultura de precisión: Conjunto de tecnologías aplicadas al trabajo del campo con la finalidad de reunir información para comprender distintos factores de la agricultura.

Series temporales: Es un conjunto de observaciones que se miden durante un lapso de tiempo ordenados de manera cronológica, los cuales pueden estar distribuidos en intervalos iguales o desiguales.

Predicción: También es conocido como pronóstico, el cual involucra una suposición sobre el futuro, generalmente, en base a un conjunto de hechos o datos observados.

Toma de decisiones: Es un componente vital de cara a la gestión de una organización que consiste en aplicar acciones en base a información analizada.

Integración de datos: Consiste en combinar o relacionar datos de distintas fuentes brindando un acceso único a través de una vista, es un proceso frecuente en el campo de Big Data.

Datos en streaming: Hace referencia al envío continuo de datos a una alta velocidad, de manera constante y en pequeños tamaños (KB), estos tipos de datos contienen cualquier clase de información.

INTRODUCCIÓN

Para la economía de un país, la agricultura tiene una gran importancia y en el Ecuador, esta no es una excepción, más del 64% de la producción agrícola surge de los pequeños agricultores y es una de las principales fuentes de trabajo para alrededor de 1.6 millones de ecuatorianos, conforme se manifiesta en [1], la agricultura aporta un promedio de 8.5% al PIB. En el aspecto internacional, para el Ecuador, el sector agrícola conforma uno de los principales motivos de exportación, a una mayor producción, mayor es la exportación y menor es la importación. La creciente necesidad de una mayor producción requiere actualizaciones e implementación de tecnologías, algo que aún se encuentra en un proceso lento de transición para el productor ecuatoriano, generalmente por la falta de información en su implementación y los beneficios que aporta el campo de la agricultura de precisión.

En este contexto, aplicar la agricultura de precisión en el Ecuador implica fortalecer las labores agrícolas optimizando los recursos, aplicándolos en el momento y lugar adecuado, mejorar el rendimiento de los cultivos, disminuir costos a largo plazo y principalmente, mejorar la calidad de los productos.

La agricultura de precisión juega un papel fundamental de cara a la solución de problemas a largo plazo, según las estimaciones de las Naciones Unidas, la población mundial alcanzará los 9 billones de personas para el 2050, en donde, problemas como el cambio climático y el calentamiento global son cada vez más evidentes, por otro lado, los recursos renovables escasearán en mayor magnitud y específicamente, se menciona que la agricultura necesitará una mayor inversión [2].

Las tecnologías asociadas a la agricultura de precisión son el uso de sensores para la captura y acción de dispositivos, Internet de las cosas para comunicar grandes cantidades de dispositivos entre sí por medio de una red y protocolos de IoT, Big Data para el almacenamiento y procesamiento de grandes volúmenes de información, y sistemas que procesen la información aplicando técnicas de análisis de datos sobre Big Data.

El objetivo de este proyecto es aplicar integración, tratamiento y análisis a grandes volúmenes de datos de IoT pertenecientes a cultivos con la finalidad de brindar información predictiva y descriptiva con características significativas para un mejor soporte al proceso de toma de decisiones para el agricultor.

Las herramientas y tecnologías como Apache Hadoop, Apache Spark, YARN, MQTT, Apache Kafka, entre otras, serán utilizadas y explicadas a lo largo del desarrollo de este proyecto, el cual se encuentra estructurado de la siguiente manera:

Capítulo I, describe el ámbito de aplicación a través de un enfoque general del tema, además del estudio sobre las necesidades y los requerimientos para llevar a cabo este proyecto, se plantea la problemática a solucionar y la justificación que describe la importancia del tema dentro de análisis de datos sobre Big Data, la metodología y tecnologías aplicadas en la solución del tema propuesto.

Capítulo II, describe la fundamentación teórica y los procesos empleados en el desarrollo del prototipo, los objetivos a cumplir y finalmente el ensamblaje y la ejecución del prototipo.

Capítulo III, establece el plan de pruebas el cual permite verificar la calidad de los modelos de análisis aplicados y evaluación de integración de datos. De esta forma se corrobora la factibilidad de implementación de la agricultura de precisión.

1. CAPÍTULO I: DIAGNÓSTICO DE NECESIDADES Y REQUERIMIENTOS

1.1. Ámbito de aplicación

La gran demanda en el sector agrícola provoca que los controles de los factores que intervienen en el proceso de producción, se vuelvan más rigurosos y precisos [3], la integración de tecnologías modernas junto con la agricultura tradicional tiene la finalidad de elevar la producción en cuanto a cantidad y calidad. Dentro de la agricultura de precisión es indispensable la recolección de datos por medio de sensores y actuadores, el concepto de Internet de las cosas toma importancia cuando se trata de recopilar datos y controlar dispositivos a distancia [4].

Tecnologías como IoT en el Ecuador no se han optimizado por la falta de apoyo en su implementación provocado por distintos motivos, en el que destaca el poco apoyo de las entidades del estado y la poca información que se brinda sobre los beneficios y cómo realmente fortalecería aplicar agricultura de precisión, no tan solo para el agricultor, si no, para el país en general, empresas ecuatorianas con la finalidad de solventar esas deficiencias presentadas en la manera tradicional de la agricultura, poco a poco han ido modernizando sus sistemas, pero sin lograr el impacto que realmente debería tener.

1.2. Establecimiento de requerimientos

En el ámbito de la agricultura de precisión, el uso de tecnologías y medios electrónicos es importante para cada uno de los procesos requeridos ocasionando que la intervención humana sea cada vez menor y los tiempos de ejecución sean más veloces [5].

El presente proyecto se encarga de demostrar cómo implementar la agricultura de precisión, qué tecnologías y herramientas a utilizar, y cómo aprovechar los datos para un correcto análisis de cara al proceso de toma de decisiones, a continuación, se listan dichos elementos requeridos:

- Simulador de red de sensores inalámbricos
- Middleware IoT, también llamado Bridge IoT
- Protocolos de IoT, MQTT y Apache Kafka

- Bases de datos PostgreSQL y MongoDB
- Denodo como herramienta de integración y virtualización de datos
- Clúster Apache Hadoop de 3 nodos, incluido el nodo master
- Lenguajes de programación Python y R para desarrollo de scripts de extracción, carga y análisis de datos

1.3. Justificación del requerimiento a satisfacer

Como tal, la agricultura de precisión aprovecha al máximo los recursos necesarios para labores agrícolas utilizándolos en menor cantidad y obteniendo como resultado una mejor calidad de producción, por ejemplo, un gran análisis de suelo permite utilizar menos cantidad de agua, gasolina, fertilizantes, entre otros recursos, necesarios para los cultivos. Por otro lado, IoT, permite que el uso de sensores, en el caso de la agricultura de precisión, se convierta en un elemento importante para la monitorización de un cultivo. Big Data además de agilizar el almacenamiento y procesamiento de grandes volúmenes de datos, permite que estos puedan ser aprovechados y analizados para detectar patrones haciendo que la toma de decisiones sea mucho más óptima.

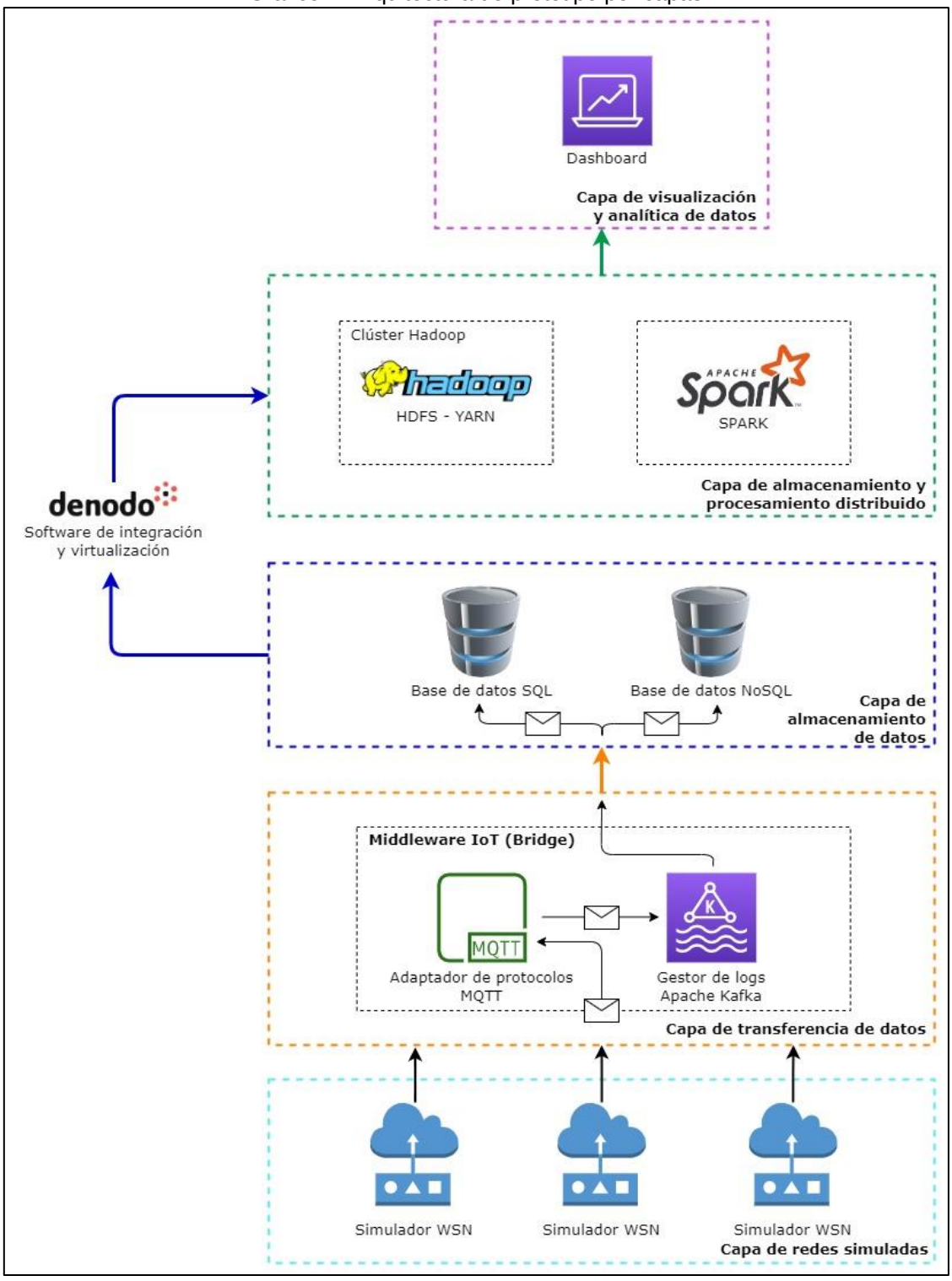
El problema que representa el análisis de datos nace desde su adquisición, muchos usuarios creen que simplemente basta con recolectar datos, pero generalmente no se aplica un tratamiento ni mucho menos, una integración para enriquecerlos, es por esto que este proyecto se enfoca en detallar el cómo aplicar cada uno de los procesos requeridos antes de emplear modelos estadísticos en la fase de análisis. Empezando desde la recolección de datos, su transferencia a gran velocidad en streaming, almacenamiento en distintas fuentes de datos, integración y virtualización de un solo conjunto de datos a partir de las distintas fuentes, almacenamiento distribuido y finalmente su análisis. Esto a largo plazo genera eficiencia en el proceso agrícola, una mejor gestión de los cultivos y principalmente una mayor calidad de producción.

2. CAPÍTULO II: DESARROLLO DEL PROTOTIPO

2.1. Definición del prototipo tecnológico

El prototipo de la propuesta tecnológica de este proyecto se conforma de una arquitectura de varias capas como se aprecia en el **Gráfico 1**:

Gráfico 1: Arquitectura de prototipo por capas



Fuente: Elaboración propia

La **capa de redes simuladas** puede estar conformada por una o varias instancias de un simulador de una red de sensores inalámbricos (WSN), que envía constantemente datos de parámetros ambientales de un cultivo.

La **capa de transferencia de datos** es la encargada de enviar los datos generados por el simulador WSN, utilizando protocolos de IoT, hacia las bases de datos SQL y NoSQL, el Bridge IoT se conforma por un adaptador de protocolos (MQTT), que a su vez se encarga de gestionar clientes suscriptores y publicadores, y por otro lado un gestor de logs (Apache Kafka), que también gestiona clientes consumidores y productores, ambos para la recepción y envío de mensajes respectivamente.

La **capa de almacenamiento de datos** se conforma por las bases de datos SQL y NoSQL, su objetivo es almacenar los mensajes que fueron receptados por el cliente consumidor del gestor de logs.

Para el proceso de análisis, no es posible tomar solamente los datos generados por los sensores, que son aquellos que se almacenan en las bases de datos NoSQL, también es necesario tomar los datos de la base de datos SQL, ya que en esta se almacenan datos que indican la fuente a la que pertenecen los mensajes almacenados en la base de datos NoSQL, a este proceso se lo denomina integración y se lo realiza mediante una herramienta de integración y virtualización que permite relacionar la información de ambas bases de datos, brindado un acceso único.

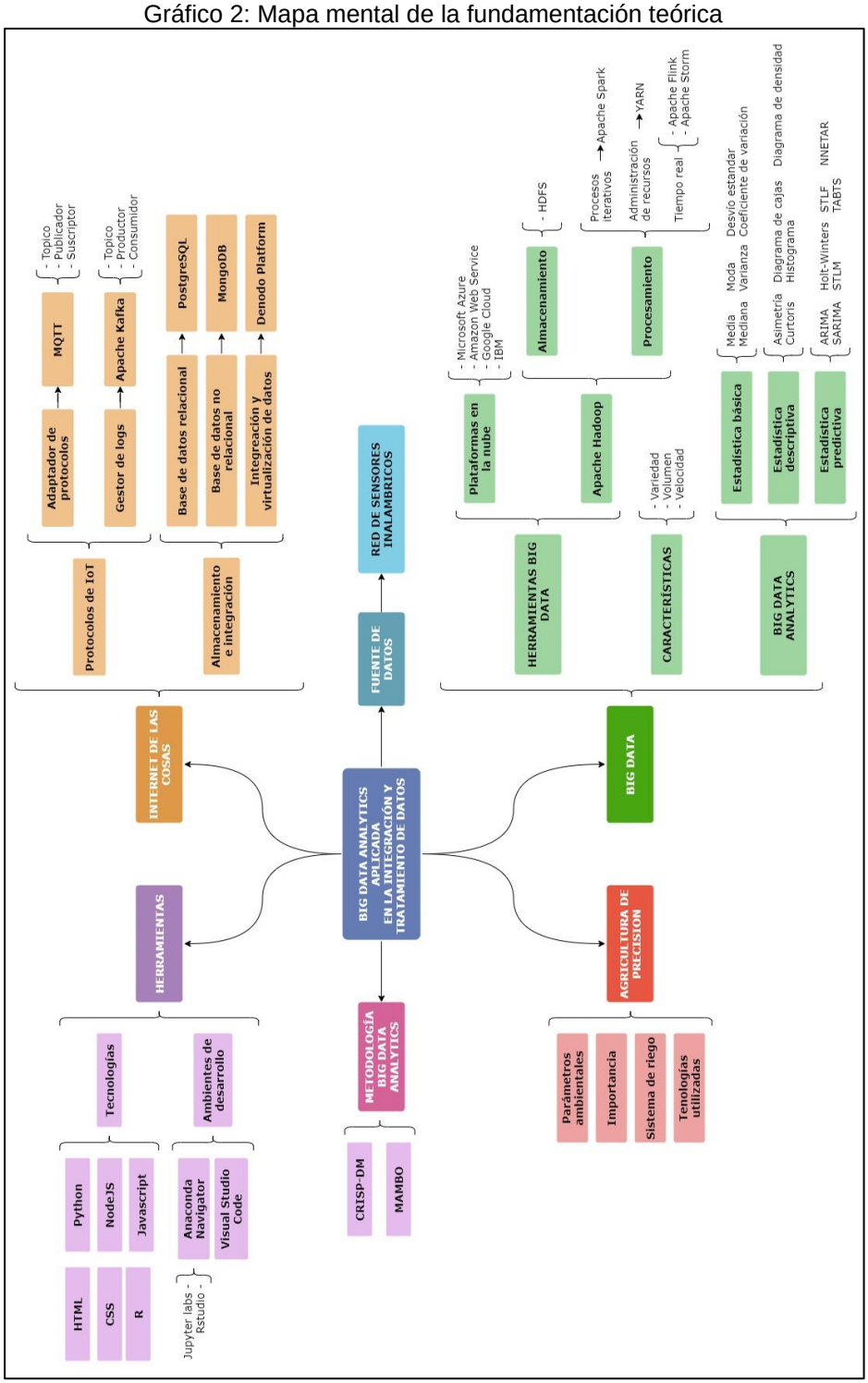
La **capa de almacenamiento y procesamiento distribuido** se conforma por un clúster Apache Hadoop de tres nodos, en donde se ejecutan las aplicaciones necesarias para el procesamiento y análisis de datos, como se mencionó anteriormente, por medio de un solo acceso a los datos virtualizados que son almacenados en el sistema de ficheros HDFS de Hadoop, de esta manera es posible aplicar técnicas de análisis sobre Big Data.

Finalmente, la **capa de visualización y analítica de datos**, presenta los resultados de los modelos estadísticos aplicados.

2.2. Fundamentación teórica del prototipo

El esquema estructurado de los temas sobre tecnologías y herramientas utilizadas para el desarrollo del prototipo se detallan en el **Gráfico 2**, esto con la finalidad de brindar un mayor entendimiento sobre su uso y la importancia de aplicarlas en este proyecto.

Gráfico 2: Mapa mental de la fundamentación teórica



Fuente: Elaboración propia

2.2.1. Agricultura de precisión

Como lo menciona [6], la agricultura de precisión (AP) es una mezcla de la agricultura tradicional y la aplicación de técnicas de ciencia y tecnología, es un concepto que administra parámetros ambientales (humedad del suelo, nivel de pH, temperatura atmosférica, humedad atmosférica, etc.) de manera precisa y automatiza algunas de las actividades que realiza un agricultor, como la planificación de riego y fertilización, entre otros.

Se basa en recopilar, procesar y analizar datos integrándolos con otra información para el apoyo a la toma de decisiones y así mejorar la producción agrícola [7] [8]. Todo el proceso mencionado es posible gracias a tecnologías como sensores, sistemas de información, maquinaria mejorada, entre otros, al final, aplicar la AP presenta beneficios como aumento de rentabilidad, mejora en cuanto a cantidad y calidad de rendimiento, reducción de costes y un menor impacto medioambiental, que es un tema muy importante en la actualidad [9].

El ciclo de la AP pasa por 4 etapas como se observa en el **Gráfico 3**:

Gráfico 3: Etapas del ciclo de la Agricultura de Precisión

Fuente: Universitat de Lleida [10]

Conforme a [10], la primera etapa es la **adquisición de datos** cuya finalidad es recolectar información de cultivos y su entorno por medio del uso de sensores, observaciones visuales y muestreos convencionales, una vez recolectado los datos, se pasa a la etapa de **extracción de información** con el objetivo de brindar información útil al agricultor como al técnico y así proceder a la etapa de **toma de decisiones**, en donde se decide sobre las tareas que hay que llevar a cabo y cómo cumplirlas, esto implica, si es necesario, mantener un manejo uniforme sobre la producción o realizar un manejo diferenciado sobre las distintas zonas, al tener la decisión tomada se continúa a la última etapa, **actuación en el campo**, cuyo objetivo es aplicar los recursos y ejercer las tareas necesarias para mejorar el proceso de producción.

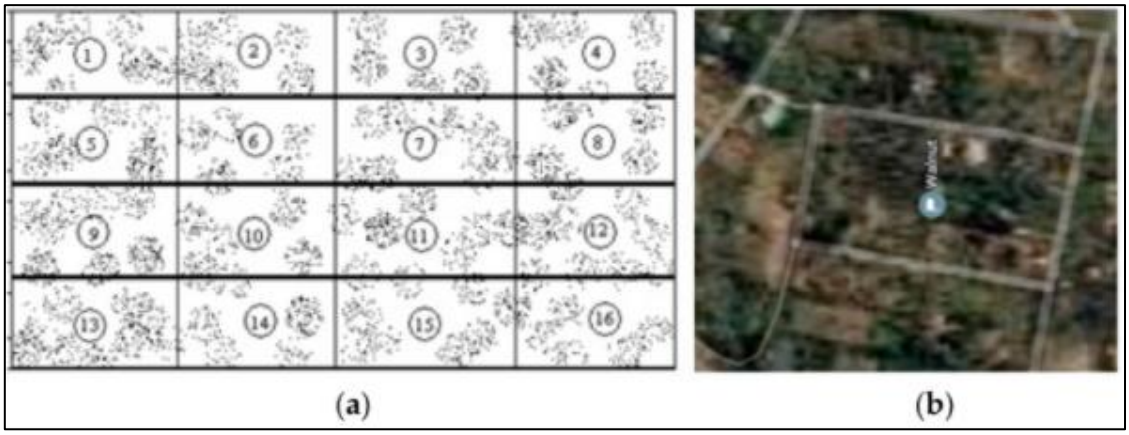
2.2.2. Sistema de riego

De acuerdo a lo establecido en [11], es la automatización de la actividad de riego con mano de obra, que tiene como finalidad, solventar el problema del riego no uniforme y el largo periodo de germinación. Existen tres tipos sistema de riego: **aspersor**, suministra gran cantidad de agua en poco tiempo y es utilizado en invernaderos con el objetivo de controlar temperatura y humedad, **goteo**, suministra el agua, dejándola caer en cantidades adecuadas utilizando una menor cantidad de agua evitando la pérdida de suelo y fertilización provocada por las altas presiones de riego, **aspersión**, utilizado principalmente en plantas de vivero para áreas extensas en donde se requiera un riego uniforme.

El sistema de riego es uno de los factores importantes para que una producción agrícola sea exitosa, además utilizando una red de sensores, es posible aprovechar las diversas fases que tiene la producción de un cultivo, todos los datos obtenidos se pueden enviar a un sistema que los almacena y analiza para apoyar a la toma de decisiones [12].

Un ejemplo de sistema de riego y su distribución de sensores en un terreno específico se aprecia en el **Gráfico 4**, los sensores se ubican cerca de la raíz del producto que se cultiva, el terreno consta de cinco tubos laterales paralelos de 20m conectados a un tubo principal, esto para proporcionar el riego de manera equitativa.

Gráfico 4: Arquitectura de sistema de riego



Fuente: Sistema de riego de precisión (PIS) [12]

Sistemas de riego inteligente

Según lo establecido por [13], con la finalidad de aumentar la productividad en el campo agrícola, nace la necesidad o urgencia de automatizar tareas que el agricultor realiza manualmente, una de esas tareas es el uso del agua, en donde, en un riego convencional son necesarios varios millones de litros de agua, algo que con un sistema de riego inteligente, tan solo se necesitarían unos pocos millones, dando paso al uso eficiente de este recurso, con el uso de tecnologías como el Bluetooth, se puede gestionar todo un sistema de riego.

2.2.3. Red de sensores inalámbricos

Una red de sensores inalámbricos (WSN) es un conjunto de nodos que funcionan de manera cooperativa como si fuesen uno solo, cada nodo se conforma de un microcontrolador, fuente de alimentación y unidad de comunicación que pueden alojar muchos sensores [14]. Los datos generados se transfieren por medio de una puerta de enlace perteneciente a la unidad de comunicación. Actualmente la WSN emerge como otro factor importante que mejora la producción agrícola, ayudando a optimizar la productividad y los recursos [15].

2.2.4. Internet de las cosas

Es un concepto que ha ido creciendo rápidamente, permite conectar miles de millones de dispositivos físicos [16], no tan solo dispositivos informáticos o dispositivos de comunicación clásica, si no, cualquier objeto que se utilice en la vida diaria para que pueda recopilar y compartir datos [17]. Todos los dispositivos están interconectados entre sí, a otros dispositivos de internet o a sistemas que

proporcionen servicios de seguridad, protección y vigilancia dentro del hogar, gestión de energía, entre otros, dependiendo del campo en donde sean aplicados [18].

Conforme a [19], dentro de la agricultura de precisión, los sistemas de apoyo aprovechan los datos recopilados por medio de la tecnología IoT para brindar servicios de pronóstico a los agricultores y productores, así la toma de decisiones se convierte en un proceso más eficiente, por ejemplo, en casos como el control de la tasa de crecimiento de cultivos, pronóstico de cambios climáticos evitar daños proactivos, control de plagas y patógenos, entre otros casos.

2.2.5. Protocolos de IoT

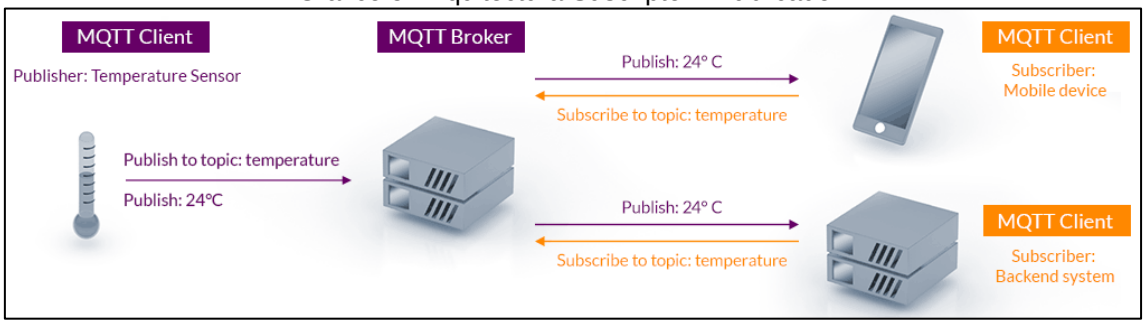
Para que los dispositivos IoT se comuniquen, es necesario el uso de protocolos de IoT, eso depende del tipo, la capa y la función que vaya a realizar el dispositivo, algunos dispositivos se conectan por medio de una puerta de enlace mientras que otros tienen la función de red integrada [20].

2.2.5.1. MQTT

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería de la capa de publicación de la familia TCP/IP, se basa en la arquitectura suscriptor-publicador, maneja información simple y liviana para el uso en equipos con restricción, bajo ancho de banda, alta latencia o redes poco confiables, su único objetivo es minimizar el ancho de banda de la red y los recursos que requiera el equipo en donde se utiliza, asegurando la fiabilidad y cierto grado de garantía al momento de entregar los mensajes [21].

Su arquitectura suscriptor-publicador (Ver **Gráfico 5**) consiste en el proceso de recepción y envió de mensajes, en primer lugar es necesario la suscripción a un tema o tópico, que es el canal de transmisión de mensajes, aquellos clientes que son publicadores y están suscritos a un tema, serán aquellos quienes podrán enviar una información y los clientes suscriptores que estén suscritos al mismo tema que los publicadores, recibirán los mensajes enviados [22].

Gráfico 5: Arquitectura Suscriptor - Publicador



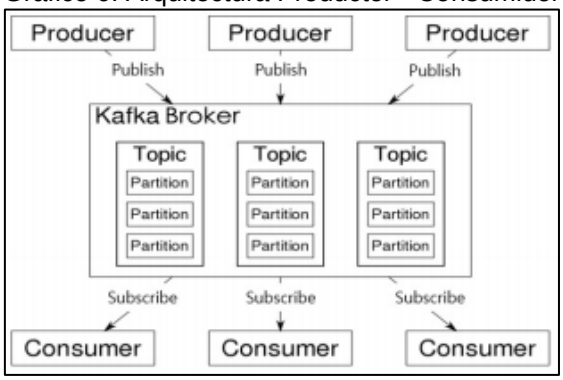
Fuente: MQTT.org [23]

2.2.5.2. Apache Kafka

Apache Kafka es un protocolo de mensajería distribuida que permite el almacenamiento de grandes volúmenes de datos que se transfieren a alta velocidad en tiempo real [24], actualmente soporta 500 mil millones de eventos por día y hasta 8 millones de eventos por segundo, Kafka garantiza la fiabilidad y rendimiento en la entrega de cada mensaje por medio de su arquitectura productor – consumidor [25].

Como se aprecia en el **Gráfico 6**, la arquitectura de Apache Kafka consta de tres elementos, un cliente productor, quien es el encargado de enviar mensajes por medio de un tópico, que es el canal de transferencia de mensajes al cual tiene que estar suscrito, y por otro lado, el cliente consumidor, quien también debe estar suscrito al mismo tópico para receptar los mensajes enviados [26].

Gráfico 6: Arquitectura Productor - Consumidor



Fuente: NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium [26]

2.2.6. Integración de datos

De acuerdo a [27], la integración de datos es la combinación de datos de distintas fuentes, es decir, un esquema de vista unificado, debido a las características de Big Data, los datos se encuentran en grandes cantidades, heterogéneos,

dinámicos y de distintas cualidades. La integración de datos puede ser por lotes, en donde la integración se realiza de manera periódica, o en tiempo real, es decir, una integración cada vez que haya un nuevo dato almacenado en cualquiera de las fuentes. En el mercado existen otro tipo de herramientas tales como IBM, Talend, Oracle, Qlik, SAS, Microsoft, entre otros.

2.2.6.1. Denodo

Denodo [28] es una plataforma de integración y virtualización de datos que ofrece un extenso catálogo para búsqueda semántica de datos y gobernanza de datos, además de consultas inteligentes impulsadas por la inteligencia artificial, infraestructura en la nube y preparación de datos para análisis.

Denodo brinda un acceso rápido a todos los datos sin importar la fuente de donde provenga, posee un gran rendimiento para el manejo de grandes volúmenes de datos, perfecto para una arquitectura Big Data.

2.2.7. Bases de datos relacionales

Una base de datos relacional [29] es un tipo de base de datos cuya finalidad es almacenar datos que poseen alguna relación con otros datos dentro de la misma, emplean el modelo relacional y están conformadas por filas y columnas, en donde las filas son registros con un ID único y las columnas son atributos que contiene cada registro.

2.2.7.1. PostgreSQL

PostgreSQL es un sistema de bases de datos relacional de código abierto más utilizado en el mundo para distintos campos como la medicina, astronomía, comercio, entre otros [30]. Su arquitectura se basa en la confiabilidad e integridad de datos para brindar soluciones innovadoras y de alto rendimiento de manera constante [31].

2.2.8. Base de datos no relacionales

Para [32], las bases de datos no relacionales forman parte de una nueva tecnología de almacenamiento alternativa que tiene la capacidad de almacenar grandes cantidades de datos sin que estos posean alguna relación entre sí, que

es una característica exigente en bases de datos relacionales. Poseen la capacidad de almacenar distintos tipos de datos, además de brindar escalabilidad y flexibilidad, haciendo que este tipo de base de datos sean una buena alternativa para el almacenamiento en redes IoT.

2.2.8.1. MongoDB

MongoDB es una base de datos NoSQL cuyo almacenamiento se basa en documentos, permite un almacenamiento en grandes cantidades, no usa tablas, en su lugar, utiliza colecciones compuestas de documentos [33]. Los documentos están conformados de pares clave-valor, que es la única unidad básica de datos que maneja MongoDB [34].

2.2.9. Big Data

Se conoce a Big Data como la colección de grandes volúmenes de datos que tiene la capacidad de crecer de manera exponencial con el tiempo, la magnitud de los datos es tan compleja que ninguna de las herramientas tradicionales de almacenamiento de información puede almacenar y procesar de manera eficiente [35]. El uso de Big Data hoy en día es muy común por la importancia de los datos para las organizaciones ya que los ayuda a crear oportunidades de crecimiento y generar cierta ventaja sobre sus rivales [36]. Para comprender de mejor manera el concepto de Big Data, existen tres propiedades definitorias que ayudan a descomponer el término de Big Data, conocidas como las 3V [37].

Según lo establecido por [38], v**olumen**, comprende la cantidad de datos que se generan y recopilan de manera constante, generando una cantidad inmensa cuya gestión y análisis se vuelve una tarea compleja. **Velocidad**, las plataformas de Big Data deben ser capaces de crear, almacenar, procesar y acceder a datos de manera rápida, y finalmente, **variedad,** que permite almacenar datos heterogéneos, es decir, datos de distinto tipo sin importar la fuente de donde provengan.

2.2.10. Herramientas Big Data: Apache Hadoop

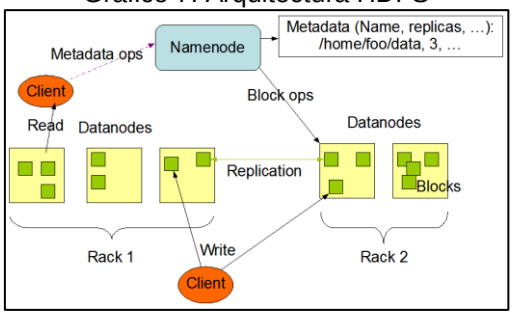
Apache Hadoop es una plataforma de código abierto que permite la gestión de procesamiento y almacenamiento de datos en el entorno de Big Data, su forma

de trabajo se basa en la distribución de grandes volúmenes de datos sobre un clúster informático, puede procesar todo tipo de datos partiendo desde un solo servidor hasta su distribución en miles de máquinas [39]. Su arquitectura se basa en maestro-esclavo, entre sus servicios principales se encuentran MapReduce, YARN y HDFS [40].

2.2.10.1. HDFS

Hadoop Distributed File System es un sistema de archivos distribuidos de código abierto utilizado en aplicaciones Big Data, es decir, aquellas aplicaciones que requieren de un acceso de alto rendimiento a grandes volúmenes de datos [41], generalmente es utilizado para escalar un clúster sencillo de Apache Hadoop, a uno más complejo conformado de muchas más máquinas [42].

Gráfico 7: Arquitectura HDFS



Fuente: Apache Hadoop [43]

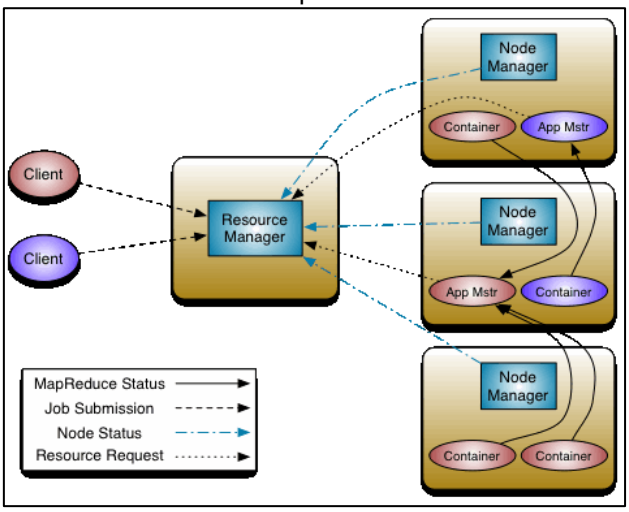
La arquitectura HDFS mostrada es el **Gráfico 7** y de acuerdo con [43], es una réplica de una arquitectura maestro-esclavo, se compone de dos elementos, **namenode**, el nodo que ejecute este software será el nodo maestro y será quien administre y ejecute operaciones del sistema de archivos, y regule el acceso a clientes, por otro lado, **datanode**, puede ser uno o varios nodos de datos que gestiona el almacenamiento realizando operaciones de lectura y escritura, creación, eliminación y replicación por bloques.

2.2.10.2. YARN

Yet Another Resource Negotiator [44] es una tecnología de gran escala destinado para las aplicaciones Big Data y administración de clústeres, nace como una característica clave en la segunda generación de Hadoop en su segmento de procesamiento distribuido, combina un administrador central de recursos con los agentes de administración de nodos para el seguimiento de las

operaciones de procesamiento de cada nodo, sus componentes principales son Resource Manager (RM), Node Manager (NM) y Application Master (Am) como se puede apreciar en el **Gráfico 8**.

Gráfico 8: Arquitectura YARN



Fuente: Apache Hadoop [45]

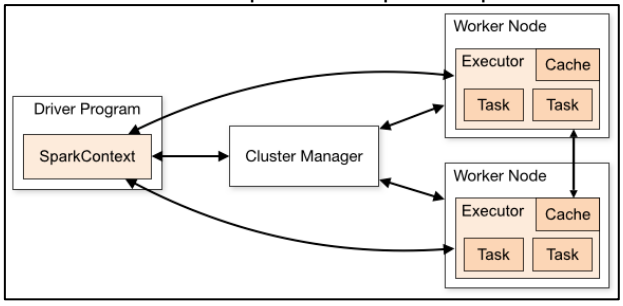
2.2.10.3. MapReduce

MapReduce [46] es un modelo de programación perteneciente a Apache Hadoop cuyo objetivo es brindar acceso a los grandes volúmenes de datos almacenados en HDFS, además agiliza el procesamiento simultáneo de tareas dentro del servidor Apache Hadoop, generalmente es utilizado para la resolución de algoritmos que requieran una ejecución en paralelo.

2.2.10.4. Apache Spark

Conforme a [47], Apache Spark es una plataforma que está destinada para el procesamiento de grandes volúmenes de datos, maximiza la ejecución de procesos en paralelo lo que hace que su velocidad de procesamiento sea más rápido que Hadoop, por otro lado posee su biblioteca de aprendizaje automático (MLlib), y permite la programación con lenguajes como Scala, Python, R y Java.

Gráfico 9: Arquitectura Apache Spark



Fuente: Apache Spark [48]

La arquitectura de Apache Spark (Ver **Gráfico 9**), cuyas aplicaciones se ejecutan como conjuntos independientes de procesos dentro de un clúster que se coordinan por medio de un objeto *SparkContext*, ejecutado en un programa principal denominado controlador, al ejecutarse el controlador, se adquieren ejecutores en los nodos esclavos del clúster, que son procesos que ejecutan los cálculos y son los encargados del almacenamiento, *SparkContext* es el encargado de distribuir las tareas a los ejecutores de los demás nodos [48].

2.2.10.5. Apache Flink

Apache Flink [49] es un marco y motor de procesamiento para realizar cálculos sobre flujo de datos ilimitados, a gran velocidad y escala dependiendo los recursos que posea el clúster ejecutor, fue pensado como una alternativa a MapReduce, posee una gran biblioteca de APIs, entre sus principales están: DataSet API y DataStream API.

2.2.10.6. Apache Storm

De acuerdo a [50], Apache Storm es un sistema computacional en tiempo real distribuido de código abierto que permite el proceso de flujo de datos, posee la misma característica de Apache Hadoop respecto al procesamiento por lotes, solo que Apache Storm lo realiza para un flujo de datos ilimitado de manera confiable, entre sus mejores características están el procesamiento de más de un millón de datos por nodo, se puede integrar con Hadoop para un mayor rendimiento y su integración con cualquier lenguaje de programación.

2.2.11. Plataformas de Big Data en la nube

Conforme a [51], gracias a la flexibilidad en la nube, hoy en día las organizaciones se permiten agregar análisis de Big Data lo que resulta algo esencial en el impulso de una infraestructura eficiente y rentable, gracias a los módulos de computación en la nube es posible potenciar soluciones de Big Data, ya que ofrecen los mismos servicios que infraestructuras locales, existen múltiples opciones como Microsoft Azure, Amazon Web Service, Google Cloud, IBM, entre otros.

2.2.11.1. Microsoft Azure

Microsoft Azure [52] es una plataforma computacional alojada en la nube, su infraestructura fue desarrollada por Microsoft con la finalidad de construir, implementar y gestionar aplicaciones y servicios por medio de un centro de datos en la nube, incluye Infraestructura como servicio (IaaS), Plataforma como servicio (PaaS) y Software como servicio (SaaS) para análisis, servicios virtuales, almacenamiento, redes, entre otros.

2.2.11.2. Amazon Web Service

Para [53], Amazon Web Service (AWS) es una plataforma en la nube que ofrece una gran cantidad de servicios a desarrolladores con la finalidad de que puedan tener acceso desde cualquier lugar y momento, las empresas utilizan AWS con la finalidad de reducir costos, aumentar su agilidad e innovar sus procesos de forma más rápida por medio de sus operaciones bajo demando con gran potencial de cómputo, almacenamiento en base de datos, entre otros servicios.

2.2.11.3. Google Cloud Platform

De acuerdo a [54], Google Cloud Platform es una plataforma de servicios en la nube que ofrece Google, entre los servicios que ofrece se encuentran la computación, almacenamiento, desarrollo de aplicaciones en la nube, redes, Big Data, aprendizaje automático, IoT, entre otros, sus principales productos son Google Compute Engine, Google App Engine, Google Cloud Storage y Google Container Engine.

2.2.11.4. IBM Cloud Platform

IBM Cloud Platform es una plataforma en la nube perteneciente a IBM que combina una Plataforma como servicio (PaaS) junto con una Infraestructura como servicio (IaaS), ofrece soporte a organizaciones y equipos de desarrollo pequeños y grandes, ofrece servicios de cálculos, máquinas virtuales, contenedores, entre otros [55].

2.2.12. Big Data Analytics

El análisis de Big Data es un proceso que consiste en examinar Big Data con la finalidad de descubrir información, patrones ocultos, correlaciones, preferencias

del cliente, tendencias de mercado, entre otros aspectos [56], toda la información encontrada será de ayuda para las organizaciones en el proceso de toma de decisiones [57]. Es un análisis avanzado que involucra un sinnúmero de aplicaciones complejas con modelos predictivos, algoritmos estadísticos, análisis hipotéticos, entre otros.

2.2.12.1. Estadística Básica

Tal como se explica en [58], la **media** es un valor que se obtiene al dividir la suma de un conjunto de números para la cantidad de dicho conjunto. Por otro lado, la **mediana** es un valor que se halla en la mitad de un conjunto de valores, el proceso de hallar ese valor, consiste en ordenar los valores de menor a mayor y determinar el valor que se encuentre en la posición de la mitad. La **moda** es aquel valor que se observa la mayor cantidad de veces dentro de un conjunto.

La **varianza** es una medida de dispersión utilizada para representar una variación dentro de un conglomerado de datos respecto a su media, se puede aplicar tanto a la población como a una muestra de los datos, mientras que el **desvío estándar** es una medida de dispersión aplicada a una distribución frecuente de datos respecto a si media aritmética, al igual que la varianza, se puede aplicar a la población como a una muestra [59]. [60] Manifiesta al coeficiente de variación como una medida que tiene la finalidad de detallar la dispersión relativa sobre un conjunto de datos, es usada para comparar un conjunto de datos de distintas poblaciones.

2.2.12.2. Estadística Descriptiva

De acuerdo a lo establecido por [61], la **asimetría** es una medida que señala la simetría de la distribución de una variable sobre su media sin recurrir al diseño de un gráfico. Mientras que **curtosis** es un valor que indica la forma de una curva de distribución, es decir, la cantidad de datos que se acercan a la media aritmética. Para [62], un **histograma** es una representación gráfica que indica la distribución de frecuencia sobre un conglomerado de datos continuos. Un **diagrama de densidad** es una representación gráfica de una versión continua y suavizada de un histograma. Por otro lado, [63] indica que, un **diagrama de**

cajas es una forma de representación gráfica de un grupo de datos por medio de sus cuartiles, muestra de forma simple, la mediana y sus cuartiles.

2.2.12.3. Estadística Predictiva

ARIMA

Para [64], es un método estadístico usado para pronosticar series temporales, netamente trabaja sobre un conjunto de datos con una estructura estándar de serie de tiempo. Es una generalización de la media móvil autorregresiva, se conforma del acrónimo:

- **Autorregresión (AR):** relación dependiente entre una observación y el número de retrasos.
- **Integrado (I):** Diferencias de observaciones para convertir la serie actual en una serie estacionaria
- **Media móvil (MA):** relación dependiente entre una observación y el error residual.

Cada componente se refleja en los parámetros del modelo, su notación estándar es ARIMA (p, d, q), en donde **p** es el número de retrasos del modelo, **d** es el número de diferencias sin procesar y **q** que indica el tamaño de la ventana de media móvil.

SARIMA

Conforme a [65], SARIMA es una extensión del modelo ARIMA que se relaciona directamente con el componente estacional de la serie de tiempo, también es conocido como ARIMA estacional, a la notación estándar ya conocida de ARIMA (p, d, q) se le agregan tres nuevos parámetros con la finalidad de indicar la autorregresión (AR), diferenciación (I) y media móvil (MA) del componente estacional, así también el parámetro que especifica el periodo de estacionalidad, de esta forma, la notación estándar es SARIMA (p, d, q) (P, D, Q) m, en donde **P** es el orden autorregresivo estacional, **D** es el orden de diferencia estacional, **Q** es el orden de media móvil estacional y **m** es el número de pasos en el tiempo de un periodo estacional.

Holt-Winters

Según [66], es un modelo que se encarga de suavizar los valores de un análisis de serie de tiempo con el objetivo de utilizarlos para predecir valores futuros, los términos clave que usa el modelo son:

- **Niveles,** indica el incremento significativo de los datos dentro de un periodo de tiempo, cada aumento significativo, indica un nuevo nivel.
- **Tendencias,** indica los patrones particulares que producen los niveles.
- **Estacionalidad,** indica el número de patrones que se repiten de manera periódica.

STLM y STLF

Ambos modelos derivan de STL, el cual es un método versátil y robusto que tiene el objetivo de descomponer series de tiempo, de ahí su acrónimo en inglés 'Seasonal and Trend decomposition using Loess', se caracteriza por manejar cualquier tipo de estacionalidad, no necesariamente con series temporales mensuales o trimestrales, permite cambios del componente estacional durante el tiempo dando la libertad al usuario de controlar la tasa de cambio y suavidad del ciclo de tendencias [67].

TBATS

De acuerdo con [68], es capaz de modelar una serie de tiempo con cualquier tipo de estacionalidad, su nombre proviene del acrónimo en inglés '**T**rigonometric seasonality, **B**ox-Cox transformation, **A**RMA errors, **T**rend and **S**easonal components.', se basa en el uso de métodos de suavizado exponencial, entre sus principales ventajas es que requiere de dos estados base independientemente del periodo y es capaz de modelar efectos estacionales de longitudes no enteras, por ejemplo, años bisiestos.

NNETAR

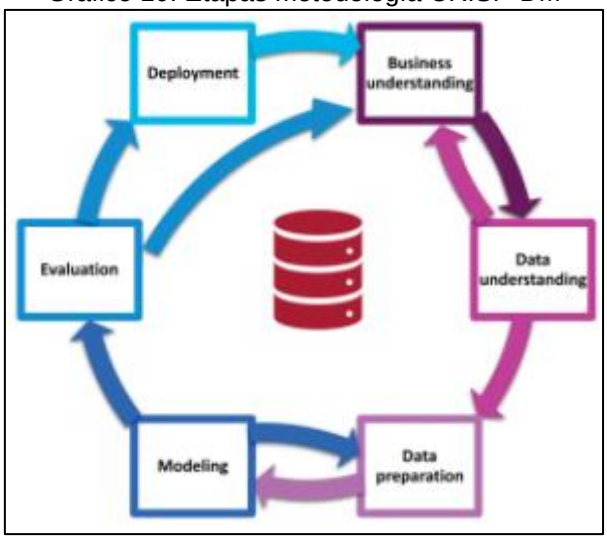
Para [69], es una red neuronal de retroalimentación de una sola capa destinada al pronóstico de series temporales univariadas, NNETAR es un paquete de pronóstico para el lenguaje R que ajusta un modelo de red neuronal a una serie temporal con retrasos.

2.2.13. Metodologías Big Data

2.2.13.1. CRISP-DM

CRoss **I**ndustry **S**tandard **P**rocess for **D**ata **M**ining (CRISP-DM) proporciona un enfoque estructurado para la planificación de proyectos destinados a la ciencia de datos o aprendizaje automático, es una metodología conocida por su robustez, aprobación, practicidad y flexibilidad [70]. Consta de 6 etapas:

Gráfico 10: Etapas metodología CRISP-DM



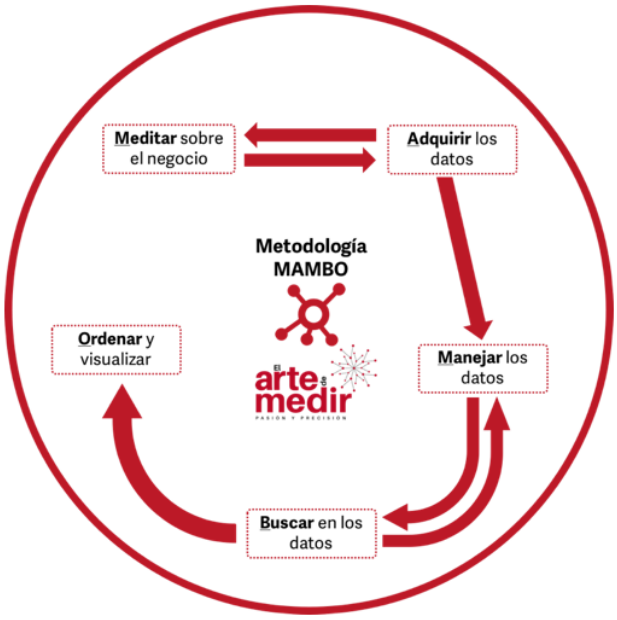
Fuente: Knowledge Discovery in Big Data from Astronomy and Earth Observation: Astrogeoinformatics [70]

De acuerdo con el **Gráfico 10** y según lo establecido en [71], la primera etapa **Business understanding** se centra en los objetivos y los requisitos del proyecto ya que permite construir los cimientos esenciales para un proyecto, su segunda etapa, **Data understanding** se enfoca en la identificación, recopilación y análisis de datos que ayuden a cumplir los objetivos propuestos, la etapa **Data preparation** tiene que la finalidad de modelar los conjuntos de datos finales a través de la selección, limpieza, construcción, integración y formateo de datos, **Modeling** es la etapa más corta ya que consiste en la construcción y evaluación de múltiples técnicas de modelado, la siguiente etapa, **Evaluation**, realiza un análisis más amplio del modelo que mejor se adapte a la organización y finalmente la etapa Deployment, se encarga de implementar el proceso de ciencia de datos y documentarlo.

2.2.13.2. Metodología MAMBO

El grupo estratégico de medición digital, El arte de medir, propone su propia metodología para proyectos de Big Data Analytics basada en CRISP-DM, consta de 5 fases denominados pasos, tal como el famosos baile.

Gráfico 11: Pasos metodología MAMBO



Fuente: El arte de medir [72]

Para [72], las etapas se definen de la siguiente manera:

Meditar sobre el negocio. Comprender los objetivos de negocio es importante para el equipo de trabajo, para ello es necesario una evaluación de los requisitos del proyecto, recursos, tecnología, tiempo de actividades, información, etc.

Adquirir los datos. Se plantea la duda sobre los datos respecto a su calidad y a si realmente pueden responder las preguntas de negocio planteadas, en este paso se aplica el proceso ETL (Extraer, Cargar y Transformar) con la finalidad de familiarizarse con los datos.

Manejar los datos. Es necesario aplicar técnicas de para el tratamiento de datos, con la finalidad de realizar una limpieza y eliminar datos que lleguen a distorsionar los resultados finales, es un proceso que realiza el equipo de Big Data.

Buscar en los datos. Comprende el análisis de los datos aplicando distintos algoritmos, técnicas o modelos con el objetivo de obtener información que sirva de apoyo durante el proceso de toma de decisiones.

Ordenar y visualizar. El último paso comprende la presentación de resultados de manera visual hacia los usuarios finales.

2.2.14.Tecnologías de desarrollo

2.2.14.1. Python

Para [73], Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, es perfecto para el desarrollo rápido de aplicaciones gracias a la combinación de sus estructuras de datos integradas de alto nivel, tipeo y enlace dinámico, además de emplearse como lenguaje de scripts para la conexión de componentes, su sintaxis es fácil de aprender ya que se enfatiza en su legibilidad. Python se basa en la modularidad y reutilización de código gracias a su gestor de paquetes. Actualmente los desarrolladores se enamoran de Python gracias a la gran productividad que proporciona.

2.2.14.2. JavaScript

De acuerdo con[74], JavaScript es un lenguaje de programación liviano, cuyo uso es generalmente destinado al desarrollo de páginas web permitiendo la interacción del usuario con el lado del cliente del sistema (frontend). Ofrece grandes ventajas sobre scripts tradiciones, por ejemplo, usar JavaScript para la validación que ingrese un usuario en los formularios de registro, también para capturar eventos que inicie el usuario, como clics, navegación de enlaces, entre otros.

2.2.14.3. Node.js

Conforme a [75], Node.js es un entorno de ejecución de JavaScript de código abierto basado en el motor JavaScript V8 de Chrome, permite el desarrollo de aplicaciones web de forma rápida y escalable. Utiliza el modelo E/S sin bloqueo controlado por eventos con características que lo hacen liviano, eficiente y perfecto para aplicaciones en tiempo real con uso intenso de datos que se ejecuten en dispositivos compartidos.

2.2.15. Ambientes de desarrollo

2.2.15.1. Anaconda Navigator

Según [76], Anaconda Navigator es una interfaz gráfica de escritorio perteneciente a la distribución Anaconda de Python, conformada por una gran cantidad de aplicaciones y paquetes, permite la gestión de paquetes, entornos y canales sin la necesidad de usar una línea de comandos, disponible para los sistemas operativos Windows, macOS y Linux.

2.2.15.2. Jupyter Notebook

Es una aplicación de código abierto con la finalidad de crear y compartir documentos web en formato JSON, admite lenguajes de programación como Julia, Python y R. Su mantenimiento es realizado por el grupo Project Jupyter. Está compuesto por un conjunto de núcleos, los cuales cada uno es un motor de ejecución para el lenguaje seleccionado, encargado de procesar solicitudes y retornar las respuestas correspondientes. Su kernel por defecto es IPython [77].

2.3. Objetivos del prototipo

2.3.1. Objetivo General

Realizar análisis de grandes volúmenes de datos de IoT, caso de uso de agricultura de precisión, aplicando procesos de recolección de datos de redes de sensores, integración y transformación a nivel de Cloud Computing, para la obtención de resultados predictivos y descriptivos que ayudan en la toma de decisiones.

2.3.2. Objetivos Específicos

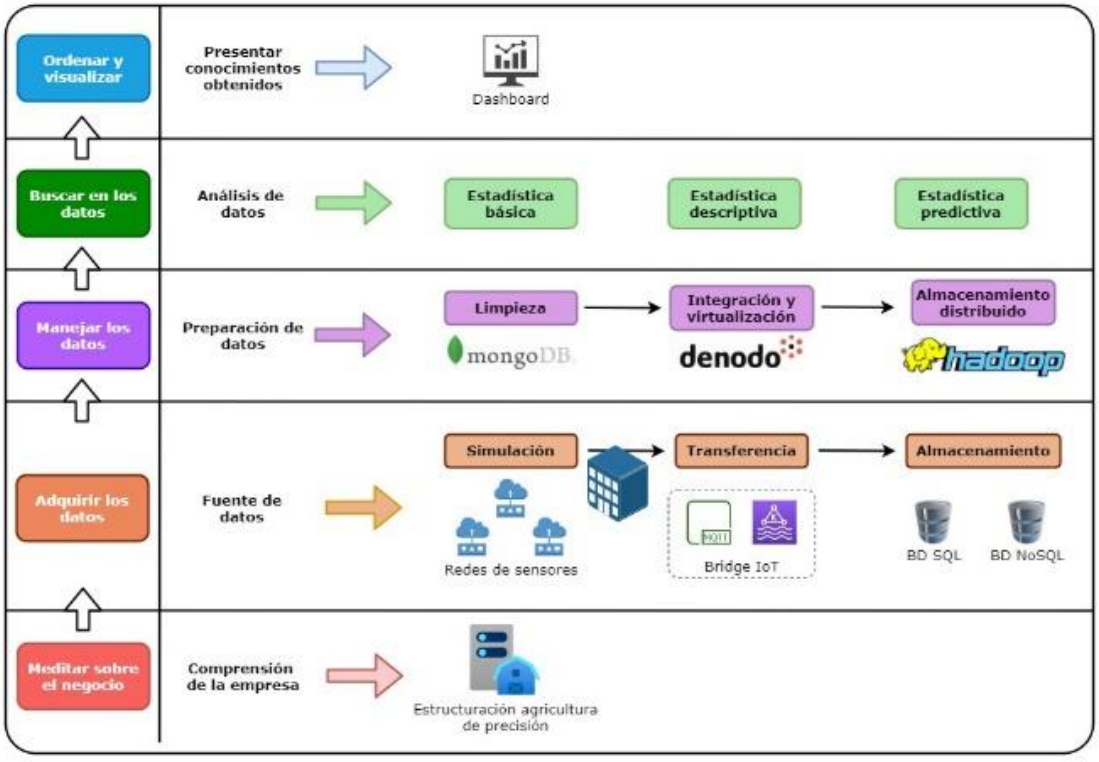
- Generar datos por medio de un sistema de simulación de una red de sensores inalámbricos (WSN) de un sistema de riego.
- Enviar los datos generados de la WSN a la Cloud Computing mediante protocolos de IoT (MQTT y Apache Kafka).
- Desarrollar un middleware IoT para la transferencia de datos desde los protocolos IoT hasta las bases de datos.
- Aplicar proceso de integración de datos para crear una base de datos lógica que sirva como punto de acceso único.

- Implementar un clúster Hadoop que almacene datos de IoT preparados para el análisis y visualización.
- Transformar y cargar datos de IoT al clúster Hadoop, enriquecidos con datos empresariales de agricultura de precisión.
- Implementar y validar modelos de minería de datos aplicados a datasets de IoT.
- Implementar un dashboard que presente gráficos estadísticos y resultados analíticos que ayuden a la toma de decisiones.

2.4. Diseño del prototipo TCI

Para el desarrollo de este proyecto se ha tomado en cuenta los pasos de la metodología MAMBO en combinación con ciertas etapas del ciclo de vida de Big Data y actividades de consideración propia para encajar cada una de las herramientas y tecnologías utilizadas en el proyecto. El **Gráfico 12** muestra a detalle las etapas por cada paso de la metodología.

Gráfico 12: Pasos metodología MAMBO



Fuente: Elaboración propia

2.4.1. Paso 1: Meditar sobre el negocio

Los sistemas de agricultura de precisión utilizan información de factores ambientales como la humedad y temperatura del ambiente, humedad del suelo, velocidad y dirección del viento, radiación solar, entre otros, para analizar la información de cultivos, brindar apoyo en la toma de decisiones y afectar de manera positiva el proceso agrícola de un producto.

Todos los datos de los factores ambientales son recopilados por medio de sensores a través de un dispositivo Arduino que se comunica a un protocolo de comunicación.

Para este proyecto, la información a tomar en cuenta para la integración, tratamiento y análisis de los datos es la de dispositivos a los cuales se conecta una red de sensores, información de los sensores y los pines de conexión, datos importantes de la empresa y la zona en la que se ubica la parcela de un determinado cultivo, también se ha considerado información del tipo de suelo y riego que se aplica. Toda la información mencionada dentro de la agricultura de precisión resulta importante para mantener una buena administración.

2.4.2. Paso 2: Adquirir los datos

El proceso de adquisición de datos se conforma de tres etapas: simulación, transferencia y almacenamiento.

La etapa de **simulación** se basa en la ejecución de un simulador WSN desarrollado en NodeJS que permita la conexión a un servidor MQTT, y la creación y envío de dos tipos de mensajes, configuración y lecturas, los mensajes de tipo configuración son aquellos que se almacenan en la base de datos SQL (PostgreSQL) por ser un tipo de mensaje que no se envía frecuentemente y cuyos datos necesitan estar relacionados, y los mensajes de tipo lecturas, son aquellos que corresponden a los valores generados por los sensores y son almacenados en la base de datos NoSQL (MongoDB) por ser un tipo de mensaje que se envía de forma masiva. Los mensajes generados por el simulador WSN están formato JSON.

Gráfico 13: Estructura de mensajes de configuración (izq) y lecturas (der)

```
{
    "nombre" : "Texto",
    "empresa" : "Texto",
    "descripcion" : "Texto",
    "localizacion" : "Texto",
    "mac" : "Texto",
    "datos" : [
        {
            "categoria" : "Texto",
            "senial" : "Texto",
            "interfaz" : "Texto",
            "ted_id" : Numérico
        }
    ]
}
```

```
{
    "mac" : "Texto",
    "datos" : [
        {
            "interfaz" : "Texto",
            "valor" : Número
        }
    ],
    "paquete_id" : "Texto",
    "estado" : Boolean,
    "bateria" : Numérico,
    "fecha" : "Texto YYYY-M-d h:m:s:ms"
}
```

Fuente: Elaboración propia

La etapa de **transferencia** se conforma de un middleware IoT, desarrollado en NodeJS, que permite el envío de mensajes a través de los protocolos de IoT, MQTT y Apache Kafka, finalmente, la etapa de **almacenamiento**, como su nombre lo indica, se basa en dirigir los mensajes que llegan al servidor Apache Kafka a su respectiva base de datos.

2.4.3. Paso 3: Manejar los datos

Este paso se basa en la preparación de los datos antes de su análisis, consta de 3 etapas: limpieza, integración y virtualización, y almacenamiento distribuido.

Los mensajes de lectura que genera y envía el simulador, tienen una frecuencia de envío de 300 ms, considerando los recursos de la máquina en donde se ejecuta, se da la situación de envío de mensajes repetidos, eso para el análisis sería algo contraproducente, la etapa de **limpieza**, se basa en aplicar un algoritmo que detecte y elimine aquellos mensajes que tienen la misma información dejando un único mensaje, por medio de comandos en MongoDB.

El tener simplemente almacenada la información de las lecturas no sirve de mucho para el análisis de información, hay que recordar que los mensajes de lecturas simplemente están compuestos de identificadores del dispositivo que los genera, los pines de conexión y los valores de los sensores, es decir, son datos en bruto, que no tienen sentido alguno, para ello es necesario la integración de la información almacenada en MongoDB y PostgreSQL, la etapa de **integración y virtualización** es la encargada de relacionar los datos de las distintas bases de datos para armar un solo conjunto, se lo realiza utilizando el software Denodo Platform, finalmente, la etapa de **almacenamiento distribuido** consiste en utilizar un clúster y por medio de un script en Python, almacenar la

vista generada en Denodo, porque los datos generados por Denodo se tratan de datos virtualizados, es decir, no están alojados en un espacio físico, para lo cual se utiliza HDFS, una herramienta necesaria para el almacenamiento de grandes volúmenes de datos.

2.4.4. Paso 4: Buscar en los datos

Durante este paso, se aplican las técnicas de análisis sobre el conjunto de datos almacenado en el clúster, es decir, los datos que están integrados desde las bases de datos PostgreSQL y MongoDB como un solo conjunto de datos, a continuación, se detallan las técnicas aplicadas:

Tabla 1: Técnicas de análisis de datos

Tipo de análisis	Técnica	
Estadística básica	Media	
	Mediana	
	Moda	
	Varianza	
	Desvío estándar	
	Coeficiente de variación	
Estadística descriptiva	Asimetría de Pearson	
	Curtosis	
	Diagrama de cajas	
	Histograma	
	Diagrama de densidad	
Estadística predictiva	Series temporales	ARIMA
		SARIMA
		Holt-Winters
		STLM
		STLF
		NNETAR
		TBATS

Fuente: Elaboración propia

2.4.5. Paso 5: Ordenar y visualizar

Finalmente, es necesario presentar la información de una manera sencilla y comprensible para el usuario final, a través de un dashboard con la finalidad de aportar información para la toma de decisiones.

2.5. Ejecución y/o ensamblaje del prototipo

Los requisitos de hardware empleados en este proyecto son los siguientes:

Tabla 2: Requisitos de hardware

Sistema Operativo	Windows 10
Arquitectura	64 bits
Procesador	Intel Core i5-4440 3.10 GHz
Memoria RAM	24 GB
Tarjeta gráfica	NVidia GTX 750Ti 2GB
Almacenamiento	1 TB

Fuente: Elaboración propia

2.5.1. Simulación, transferencia y almacenamiento de datos

2.5.1.1. Simulación

Se ha utilizado un simulador de sensores elaborado por estudiantes y docentes del grupo de investigación AUTOMATHTIC de las carreras de Ingeniería de Sistemas y Tecnologías de la Información de la Universidad Técnica de Machala, entre sus funciones más importantes se encuentran:

Algoritmo 1: Conexión a MQTT

```
var client = mqtt.connect('mqtt://' + conexion.host + ":" +
conexion.puerto, {
    username: conexion.usuario,
    password: conexion.pass
});

client.on('connect', function () {
    console.log('se conecto al mqtt');

    client.subscribe(topicos.topico, function () {
        console.log('se suscribió al topico ' + topicos.topico);
    });
}
```

Fuente: Elaboración propia

Crear un cliente MQTT y suscribirse a un tópico permite que se envíen los datos utilizando el protocolo, para ello es necesario conocer el host, puerto, usuario y contraseña del servidor MQTT.

Algoritmo 2: Generar y enviar mensajes

```
let dispJSON = {
    mac: dispositivo.mac,
    datos: [],
    paquete_id: "100",
    estado: false,
    bateria: num_al(0, 5)
};

let configJSON =
JSON.parse(JSON.stringify(configuracionQueryResult));

dispJSON.fecha = GetDateAndHour(600000  , ultimaLectura[0].fecha);
configJSON.datos.forEach(async dato => {
    if (dispositivo.datos[indice].indexOf('out') == (-1)) {
        if (dispositivo.datos[indice].indexOf('a') == (-1)) {
            dispJSON.datos.push({
                interfaz: dispositivo.datos[indice],
                valor: num_al(0, 1)
            });
        } else {
            dispJSON.datos.push({
                interfaz: dispositivo.datos[indice],
                valor: generarValorTed(dato.ted_id, dispJSON.fecha),
            });
        }
        indice++;
    } else {
        indice++;
    }
});

client.publish(topicos.topico, JSON.stringify(dispJSON));
io.sockets.emit('vision', dispJSON);
```

Fuente: Elaboración propia

Es importante generar la estructura de los mensajes a enviar, para ello se necesitan los objetos de JavaScript, al ser un simulador, es necesario manejar una frecuencia de envío distinta al tiempo real de ejecución, por eso dentro de la estructura de mensaje, en el campo fecha, por cada envío, el siguiente tendrá un aumento en el tiempo de 10 minutos, luego simplemente, se identifica el tipo de sensor y su señal, si es digital, genera valores entre cero y uno, y si es analógico, genera valores en base al tipo de sensor, finalmente, cuando la estructura del mensaje esté completa, se lo envía al servidor MQTT.

2.5.1.2. Transferencia

Para la transferencia fue necesario el desarrollo de un middleware IoT, también llamado Bridge IoT, en base al trabajo de titulación elaborado por Kevin Valarezo [78], estudiante de la carrera de Ingeniería de Sistemas de la Universidad Técnica de Machala, para comunicar los protocolos de IoT, MQTT y Apache Kafka.

Algoritmo 3: Manejador de clientes MQTT

```
conectar(callback) {
    this.clienteMqtt = mqtt.connect(`mqtt:

    this.clienteMqtt.on('error', (err) => {
        return callback(err);
    })

    this.clienteMqtt.on('connect', () => {
        console.log(`Cliente MQTT
${this.datosConexion.clt_con_host}:${this.datosConexion.clt_con_puer
to} conectado`);
    })

    this.clienteMqtt.subscribe(this.listaTopicos);

    this.clienteMqtt.on('message', function (topico, mensaje) {
        return callback(mensaje, topico);
    });
}
```

Fuente: Elaboración propia

Los clientes MQTT se gestionan por medio de este manejador que además de su conexión, permite el envío y recepción de mensajes como publicadores y suscriptores respectivamente.

De igual manera, es importante tener un manejador de conexiones al servidor Apache Kafka, que además de la conexión, permita la creación de consumidores y productores para la recepción y envió de mensajes respectivamente.

Algoritmo 4: Manejador de clientes Apache Kafka

```
async conectarProductorConsumidor() {
    this.clienteKafka = new Kafka({
        clientId: 'kafka_master_id',
        brokers:
[`${this.datosConexion.clt_con_host}:${this.datosConexion.clt_con_pu
erto}`],
        logLevel: logLevel.ERROR
    })

    this.productor = this.clienteKafka.producer();
    this.consumidor = this.clienteKafka.consumer({ groupId:
'kafka_productor_master', sessionTimeout: 7000});
    this.consumidor.logger().setLogLevel(logLevel.INFO)

    try {
        await this.productor.connect();
        await this.consumidor.connect();
        this.suscribirTopicos(this.listaTopicos);
        console.log(`Productor y consumidor de cliente Kafka
${this.datosConexion.clt_con_host}:${this.datosConexion.clt_con_puer
to} conectados`);
    } catch (error) {
        console.log(error);
    }
}
```

Fuente: Elaboración propia

Algoritmo 5: Transferencia de mensajes de lectura

```
clienteMqtt.conectar((mensaje, topico) => {
    console.log('************************* SUSCRIPTOR MQTT *************************');
    console.log(`Topico: ${topico}`);
    console.log(JSON.parse(mensaje.toString()));
    let topicoKafka = this.convertirTopico(topico);
    clienteKafka.enviarMensaje(mensaje.toString(), topicoKafka);
});

setTimeout(() => {
    clienteKafka.lecturas(async (topico, mensaje) => {
        let jsonMensaje = JSON.parse(mensaje.value.toString());
        console.log('************************* CONSUMIDOR KAFKA *************************');
        console.log(`Topico: ${topico}`);
        console.log(jsonMensaje);
        console.log();

        try {
            if (topico === 'lecturas') lecturasDAO.guardarLecturaMongo(jsonMensaje)
        } catch (error) {
            console.log(error);
        }
    })
}, 1000);
```

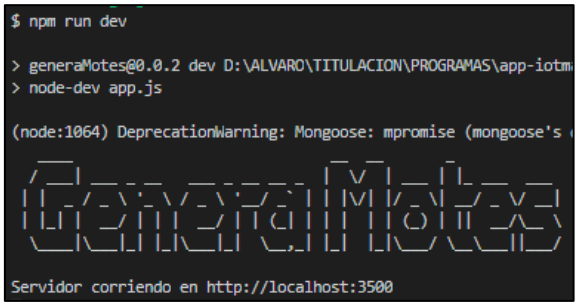
Fuente: Elaboración propia

El proceso de transferencia empieza por la creación y conexión de clientes MQTT y Apache Kafka, el cliente MQTT como suscriptor, estará esperando los mensajes enviados por el simulador e inmediatamente el cliente Apache Kafka como productor enviará los mensajes hacia el servidor Apache Kafka, a su vez, el mismo cliente como consumidor estará a la espera de los mensajes para determinar si son configuraciones o lecturas, si son lecturas, se procede a almacenar el contenido de los mensajes en MongoDB, caso contrario, al ser una configuración se procede a guardar el contenido del mensaje en PostgreSQL, validando que la información a guardar no se encuentre ya registrada con anterioridad.

El modo de ejecutar las aplicaciones para que se pueda realizar la transferencia, es mantener levantados los dos servidores, el simulador y el Bridge IoT, para este proyecto, el simulador y el Bridge IoT se encuentran funcionando en los puertos 3500 y 4000 respectivamente.

Al ejecutar el comando *npm run dev* en una terminal dentro del directorio del simulador, se ejecuta el servidor y se muestra la información del host en donde está corriendo.

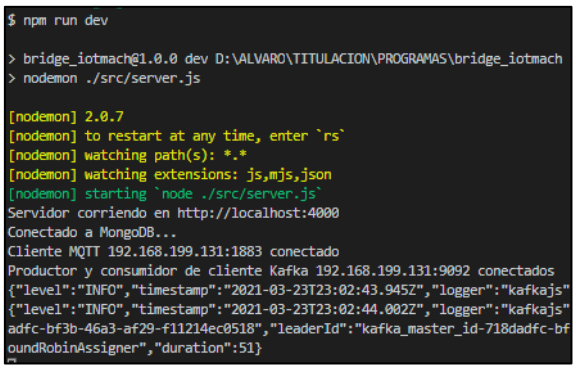
Gráfico 14: Ejecución - Simulador



Fuente: Simulador WSN

De igual manera, se emplea el mismo comando para ejecutar el Bridge IoT en donde al levantarse, se observa información del host en donde está corriendo, los clientes MQTT y Apache Kafka a los que se conecta, y mensajes de información como respuesta de parte del servidor Apache Kafka.

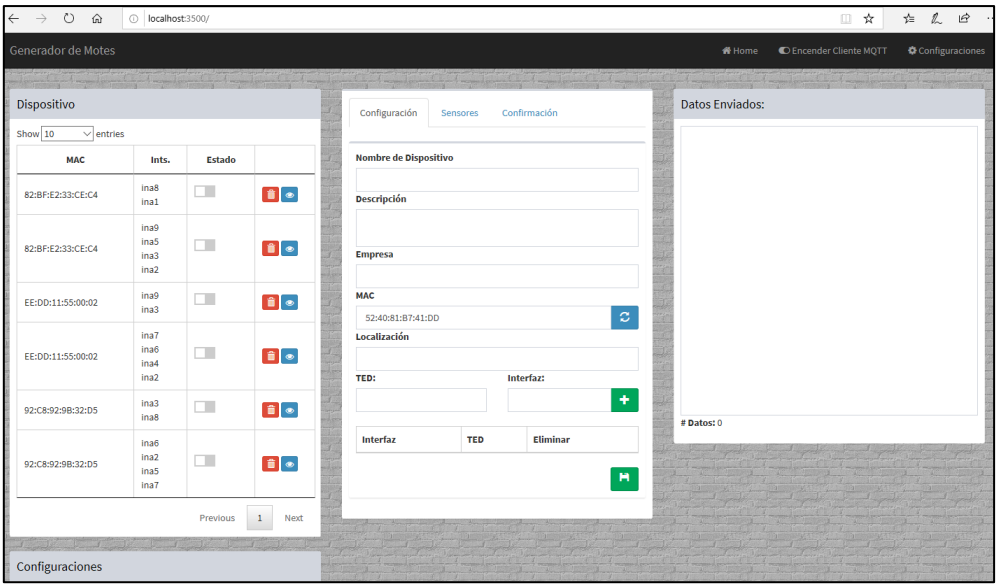
Gráfico 15: Ejecución - Bridge IoT



Fuente: Bridge IoT

Para enviar mensajes hacia las bases de datos, ingresar al host del simulador en un navegador, *http://localhost:3500*.

Gráfico 16: Simulador desde navegador

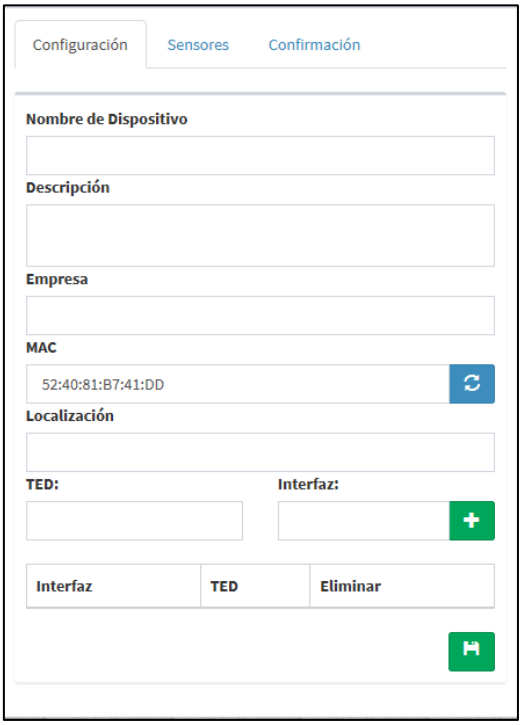


Fuente: Simulador WSN

Envío de mensajes de configuración

Los mensajes de configuración permiten añadir sensores a un dispositivo existente o añadir un nuevo dispositivo con sus sensores, para ello, se debe llenar la información del formulario mostrado a continuación.

Gráfico 17: Formulario para mensajes de configuración



Fuente: Simulador WSN

Se necesitan llenar todos los campos, al ser un simulador, se establece la opción de generar una MAC aleatoria o colocar una manualmente, para los Ted's, es necesario colocar su código.

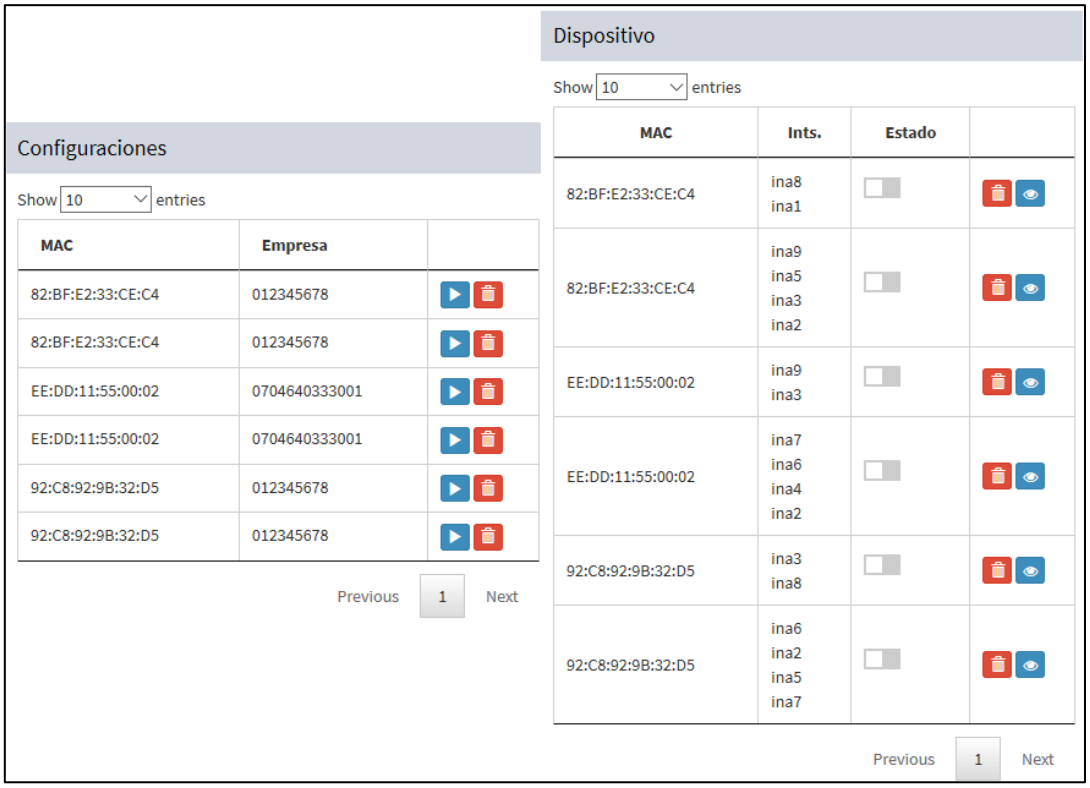
Tabla 3: Códigos Ted's para simulador

Código	TED
16	Pluviómetro
17	Velocidad del viento
18	Humedad del suelo
19	Temperatura relativa del aire
21	Humedad relativa del aire
22	Temperatura del suelo
23	pH del suelo
24	Electro conductividad

Fuente: Elaboración propia

Al crearse la configuración, se despliega al lado izquierdo, las configuraciones y los sensores correspondientes a los dispositivos.

Gráfico 18: Lista de configuraciones, dispositivos y sensores creados



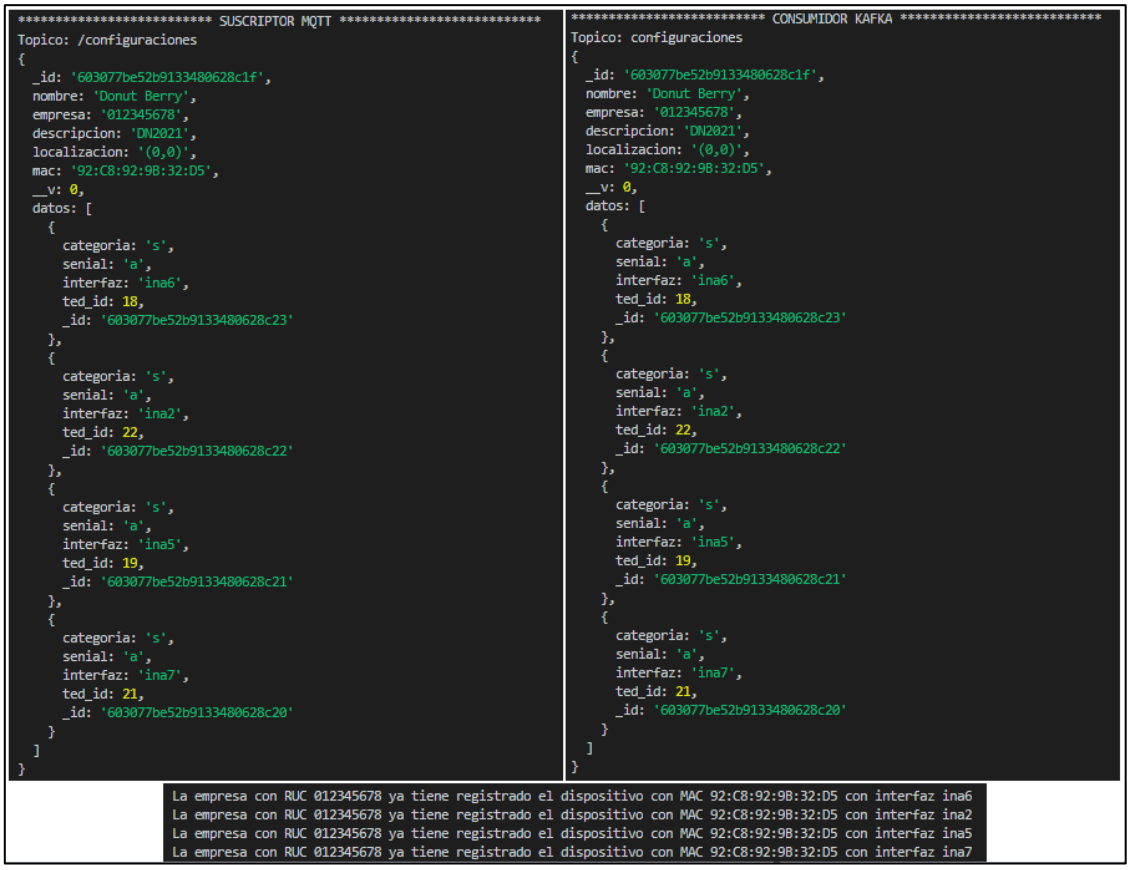
Fuente: Simulador WSN

El envío de mensajes de configuración es manual, como se mencionó anteriormente, es un tipo de mensaje que no se envía con frecuencia. En la lista de configuraciones se observa un botón de color azul que permite el envío de mensaje de configuración, al presionarlo, instantáneamente el Bridge IoT recibe el mensaje y lo almacena en la base de datos SQL.

Como se observa en el **Gráfico 19**, el Bridge IoT recibe primero el mensaje en su suscriptor MQTT para luego enviarlo por medio de un productor Apache Kafka al servidor del mismo nombre, y finalmente, se observa que el mismo mensaje, con el mismo contenido es receptado por un consumidor Apache Kafka para su almacenamiento en PostgreSQL.

Aquellos mensajes de configuración que contengan información ya almacenada serán notificados como existentes y no se guardarán nuevamente, caso contrario, si es información nueva de sensores para un dispositivo al igual que si es un dispositivo nuevo, se guardarán en la base de datos.

Gráfico 19: Visualización transferencia y almacenamiento de configuraciones

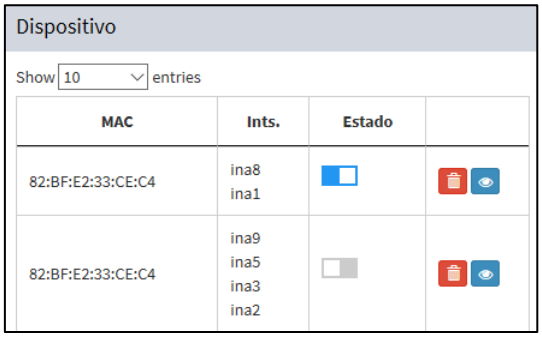


Fuente: Bridge IoT

Envío de mensajes de lectura

Los mensajes de tipo lectura son aquellos que se envían en grandes cantidades, considerando que, para aplicar técnicas de análisis de datos, es necesario generar un conjunto de datos que al menos abarque 4 años de lecturas de sensores de distintos cultivos, es por ello, que la frecuencia de envío para los mensajes de lecturas del simulador se envían con una frecuencia de 300 ms. Para iniciar el proceso de envío, se deben habilitar en la lista de dispositivos, aquellas interfaces de sensores que van a generar datos.

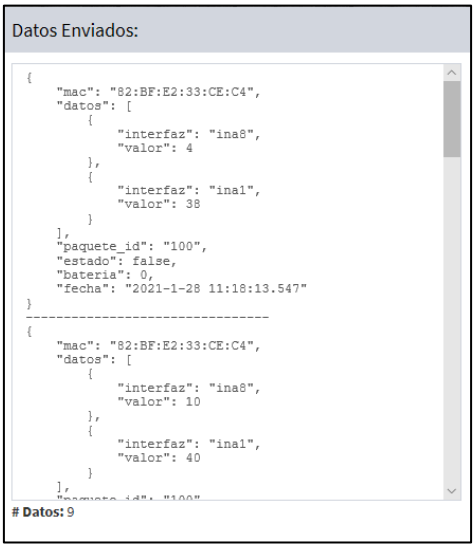
Gráfico 20: Habilitar interfaces de sensores para envío de datos



Fuente: Simulador WSN

Una vez se han habilitado las interfaces, en la parte superior se presiona el botón *Encender cliente MQTT* para iniciar el envío de mensajes a la base de datos.

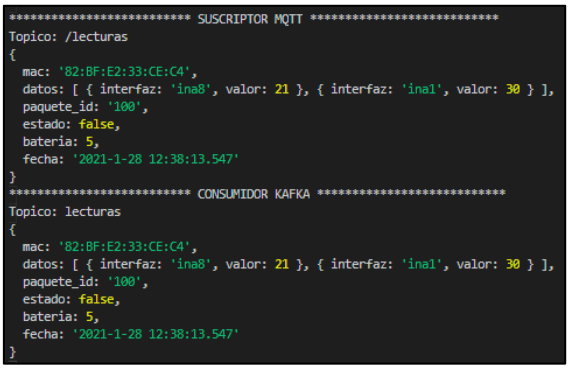
Gráfico 21: Envío de mensajes desde simulador



Fuente: Simulador WSN

Como se aprecia en el **Gráfico 21**, en un apartado ubicado en el lado derecho del navegador se observan los mensajes generados por el simulador, mientras que en el **Gráfico 22**, se observa como los mensajes llegan al suscriptor MQTT, y de igual manera que con los mensajes de configuraciones, son enviados por un productor Apache Kafka, recibidos por un consumidor Apache Kafka y almacenados en MongoDB.

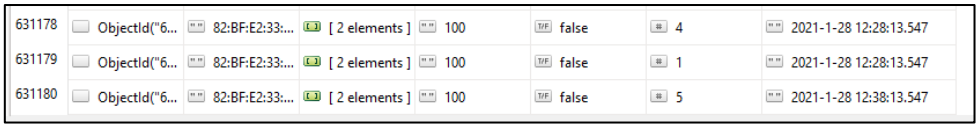
Gráfico 22: Recepción y almacenamiento de mensajes de lecturas



Fuente: Bridge IoT

Para comprobar el almacenamiento de datos, se ha utilizado el software *Robo 3T* como se observa en el **Gráfico 23**.

Gráfico 23: Comprobación mensajes almacenados en MongoDB



Fuente: Robo3T

2.5.2. Tratamiento de datos

2.5.2.1. Limpieza

Para la limpieza del conjunto de datos de este proyecto, es necesario eliminar los mensajes que se almacenaron repetidos y para ello con el mismo gestor visual para MongoDB y comandos de Mongo Shell se realiza la limpieza.

Algoritmo 6: Buscar y eliminar mensajes repetidos

```
db.getCollection('lecturas').aggregate([
    {$group: {
        _id: {fecha: "$fecha", mac : "$mac"},
        dups: {$push: "$_id"},
        count: {$sum: 1}
        }
    },
    {$match: {
        count: {"$gt": 1}
        }
    }
], { allowDiskUse: true }).forEach(function(doc) {
    doc.dups.shift();
    db.getCollection('lecturas').remove({ "_id": {"$in": doc.dups
}});
});
```

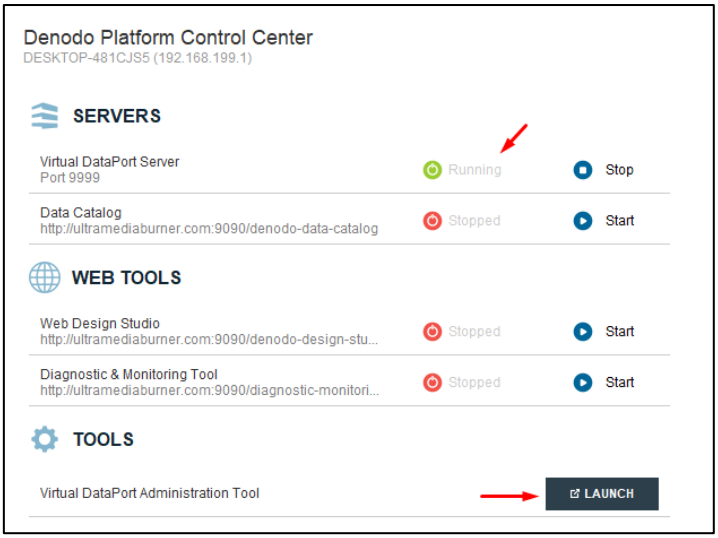
Fuente: Elaboración propia

El algoritmo consiste en agrupar y contabilizar aquellos documentos almacenados en una colección de MongoDB que presentan información duplicada a través de sus campos, *fecha* y *Mac*, una vez se han identificado los documentos duplicados se procede a removerlos dejando un único documento.

2.5.2.2. Integración y virtualización

Para integrar y virtualizar los datos desde PostgreSQL y MongoDB se utiliza Denodo, lo primero que se debe hacer es levantar el servidor e ingresar a *Denodo Virtual DataPort.*

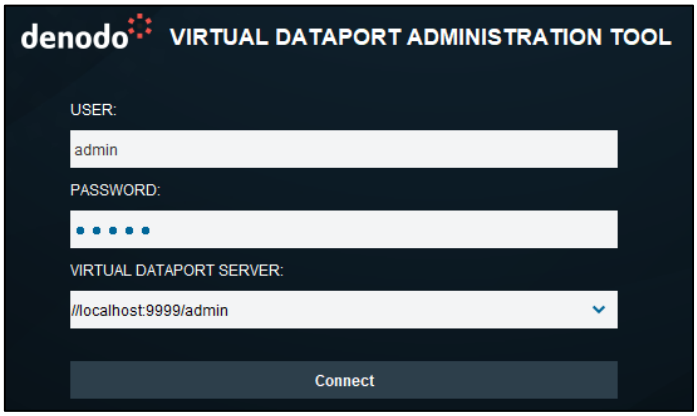
Gráfico 24: Iniciar servidor Virtual DataPort



Fuente: Denodo

Al abrir el administrador de herramientas, los datos de acceso tanto en usuario como contraseña es *admin.*

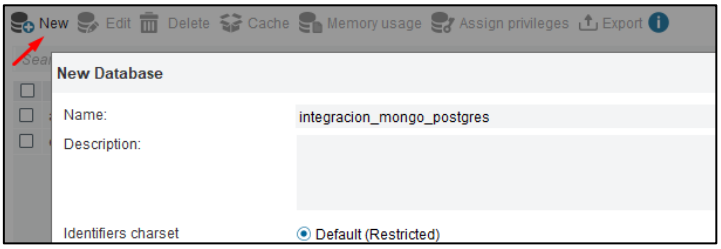
Gráfico 25: Acceso al administrador de herramientas de Virtual DataPort



Fuente: Denodo

Una vez dentro de Virtual DataPort se crea una base de datos a través de la pestaña *Administration > Database Management,* se abre un apartado en donde se observan todas las bases de datos y se presiona el botón *New* para crear una nueva, aparece un formulario en donde se coloca el nombre de la base de datos.

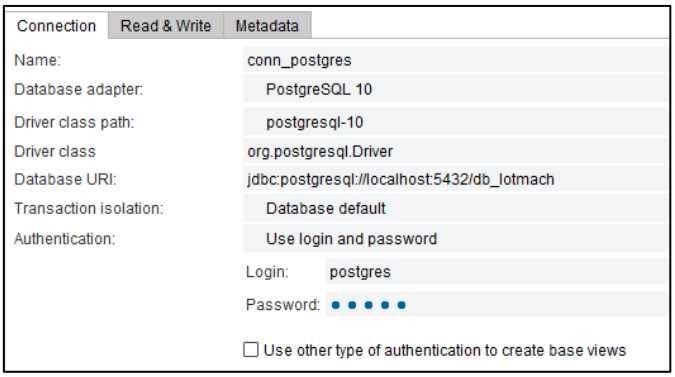
Gráfico 26: Creación nueva base de datos



Fuente: Denodo

El siguiente paso es crear las conexiones a las bases de datos PostgreSQL y MongoDB, para la conexión a PostgreSQL, clic derecho sobre la base de datos creada, *New > Data Source > JDBC*, se abre una nueva pestaña con un formulario para colocar los datos de conexión.

Gráfico 27: Formulario de conexión a PostgreSQL



Fuente: Denodo

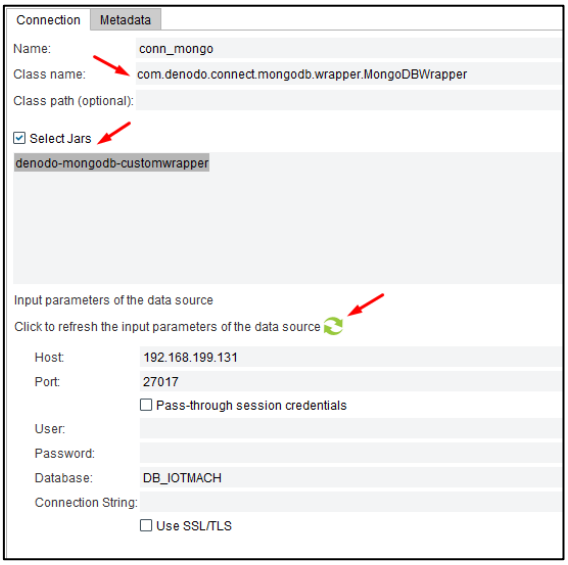
Se realiza el test de conexión, si el resultado es exitoso, se guarda la conexión. Para la conexión a MongoDB, es necesario utilizar un driver proporcionado por el soporte de Denodo, *Denodo MongoDB Custom Wrapper*, este driver se carga a través de la pestaña *File* > *Extension management*, se importa el archivo .jar que contenga dependencias, como se observa en el **Gráfico 28**.

Gráfico 28: Driver de conexión a MongoDB

Fuente: Denodo

Con el driver cargado, clic derecho sobre la base de datos, *New* > *Data Source* > *Custom*, de esta manera se carga una pestaña con el formulario para colocar los datos de conexión. Lo primero a considerar es la habilitación de la casilla de verificación *Select Jars* para seleccionar el driver que se cargó anteriormente, luego se procede a refrescar el formulario a través del botón *Refresh* como se observa en el **Gráfico 29**, se carga el formulario para los parámetros de conexión siempre y cuando el driver sea el correcto, si no, sale un mensaje de error, se llena el formulario y se guarda la conexión.

Gráfico 29: Conexión a MongoDB

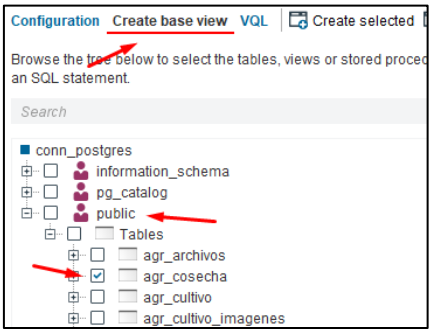


Fuente: Denodo

Al tener las conexiones a las bases de datos, lo siguiente obtener vistas de las tablas con las cuales se va a realizar la integración, primero se abre la conexión con PostgreSQL, en la parte superior de la pestaña se presiona el botón *Create*

base view, se cargan los esquemas de la base de datos, se selecciona el esquema en donde se almacenen las tablas y se procede a su selección.

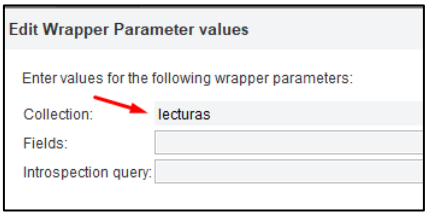
Gráfico 30: Selección de tablas PostgreSQL



Fuente: Denodo

Terminada la selección de las tablas, en la parte superior se escoge la opción *Create selected* para crear las vistas. Proceso similar se realiza para las tablas de MongoDB, se selecciona la conexión y en la parte superior derecha se encuentra la opción *Create base view*, al presionarla se abre una ventana en donde se debe ingresar la colección.

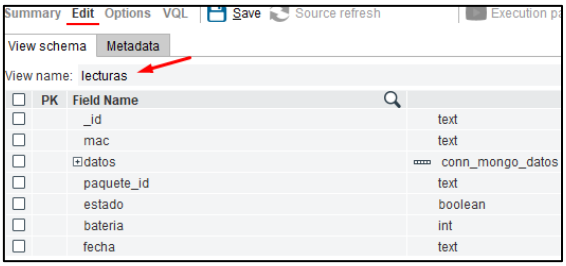
Gráfico 31: Ingreso de colección para crear vistas de MongoDB



Fuente: Denodo

Denodo genera una nueva pestaña con los campos que obtuvo de la colección seleccionada, se le da la opción de colocar un nuevo nombre a la vista y de seleccionar una llave principal.

Gráfico 32: Vista previa de vista de colección lecturas MongoDB

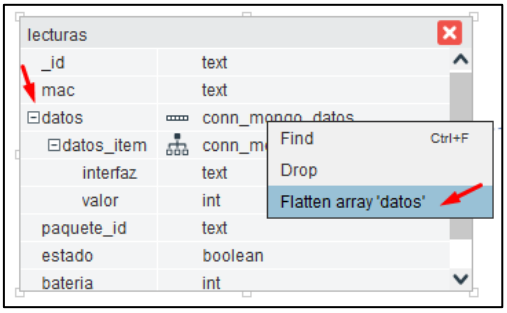


Fuente: Denodo

Al guardar la vista y ver sus datos, se observa que hay campos compuestos, es decir, constan de más variables, específicamente se habla del campo *datos*, que es un array que contiene interfaces y valores, para ello es necesario desplegar esta información, Denodo brinda una funcionalidad llamada *Flatten*, para ello, se

debe dar clic derecho sobre la vista generada, *New* > *Flatten*, se abre una pestaña en donde se despliega una tabla con la información de la vista, se debe seleccionar el campo, clic derecho y escoger la opción *Flatten array 'datos'.*

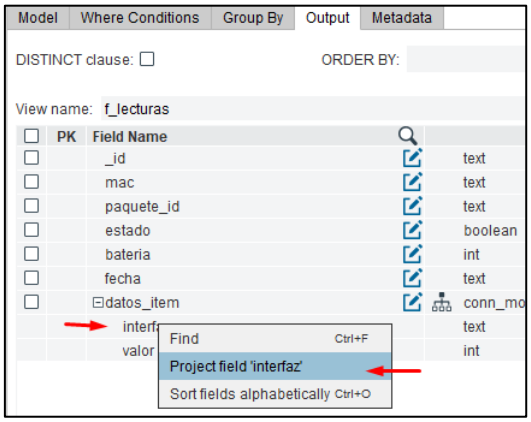
Gráfico 33: Aplicando Flatten sobre el campo datos



Fuente: Denodo

Para este caso en específico, se da la situación que los datos están contenidos además dentro de un objeto, y aún es necesario extraerlos, para ello en la pestaña *Output* se observan los campos, se debe desplegar el campo de *datos*, clic derecho sobre uno de sus ítems que contiene y seleccionar la opción *Project field,* repetir el proceso para los demás campos que son parte del campo *datos.*

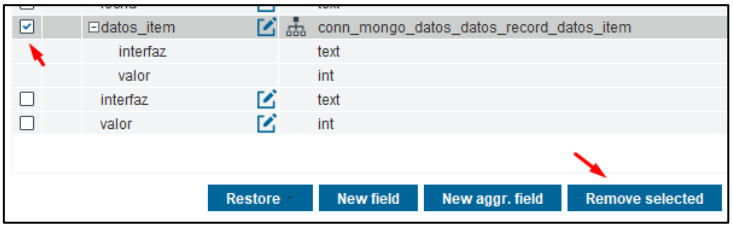
Gráfico 34: Extracción de datos del objeto campo



Fuente: Denodo

Adicionalmente, ya no es necesario tener el objeto *datos* una vez estén desplegados sus campos, se puede eliminar el campo, seleccionándolo y removiéndolo como en el **Gráfico 35**.

Gráfico 35: Eliminación de objeto datos



Fuente: Denodo

Una vez terminado todo el proceso se procede a guardar la vista, ya que eso es lo que se estaba generando cuando se aplicó la funcionalidad *Flatten*.

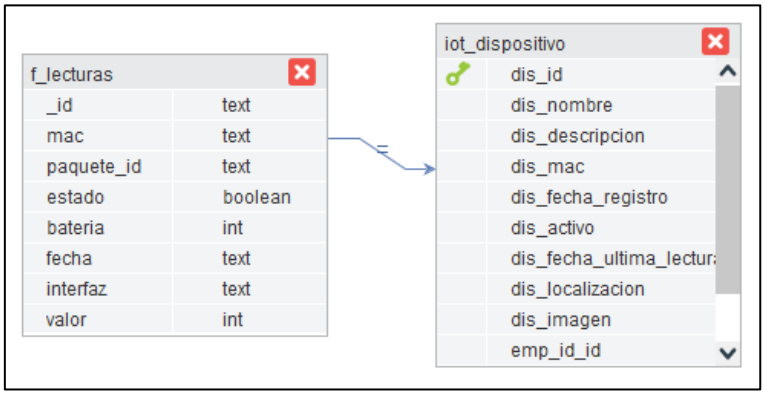
Al terminar todo el proceso de conexiones y creación de vistas, la estructura de la base de datos en Denodo, recordando que es sólo una virtualización, queda de la siguiente manera.

Gráfico 36: Estructura de base de datos en Denodo

Fuente: Denodo

Es tiempo de elaborar la vista de integración, se hace clic derecho sobre la base de datos, *New* > *Join*, el proceso es simple, se deben arrastrar las vistas de las tablas a integrar en el espacio de trabajo de la pestaña que se abre y luego unir las llaves primarias y campos que tengan relación.

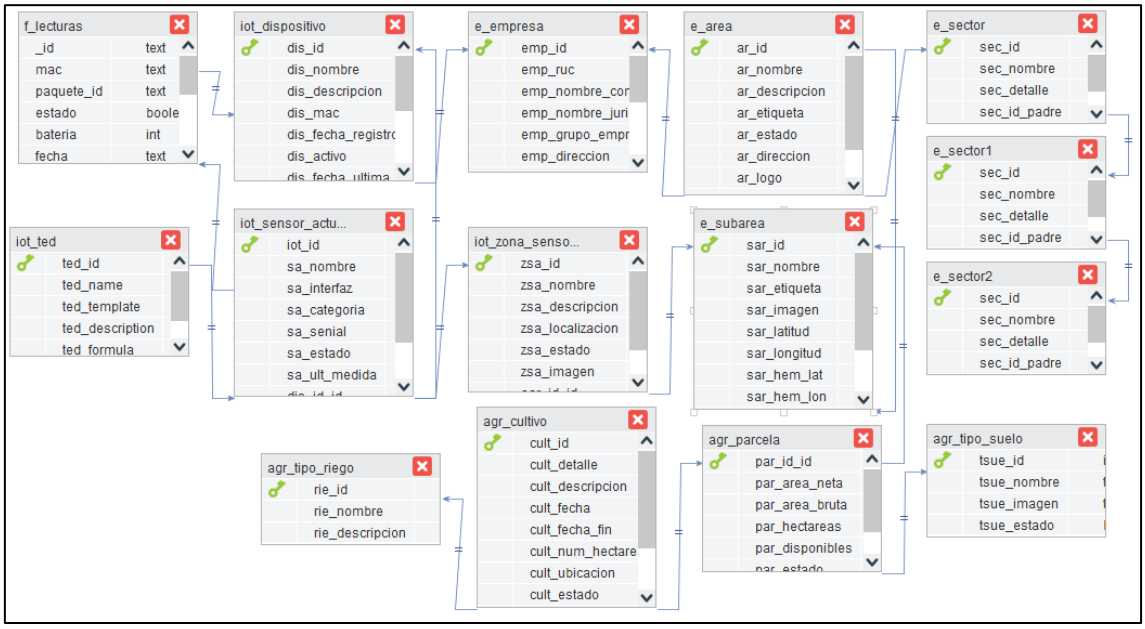
Gráfico 37: Ejemplo de integración de datos



Fuente: Denodo

El proceso se repite para cada vista necesaria a integrar, al final se obtiene una vista parecida a la que se presenta en **Gráfico 38**.

Gráfico 38: Vista de integración de datos



Fuente: Denodo

Antes de guardar la vista, es necesario eliminar aquellos campos que no serán necesarios para el análisis, así como también, de los campos duplicados, ubicarse en la pestaña *Output*, marcar los campos a eliminar y removerlos, proceso que anteriormente se hizo para la funcionalidad *Flatten*, al final, los datos que la vista presenta serán los mostrados en el **Gráfico 39**.

Gráfico 39: Selección de campos a mostrar en la vista integrada

View name: vista_db_iotmach

PK	View Name	Field Name	Field Type	Description
	f_lecturas	mac	text	
	f_lecturas	interfaz	text	
	f_lecturas	valor	int	
	f_lecturas	anio	long	
	f_lecturas	mes	long	
	f_lecturas	dia	long	
	f_lecturas	hora	time	
	iot_dispositivo	dispositivo	text	
	iot_ted	sensor	text	
	iot_zona_sensor_actuador	zona_sensor	text	
	e_empresa	ruc	text	
	e_empresa	empresa_comercial	text	
	e_subarea	subarea	text	
	e_area	area	text	
	e_sector	ciudad	text	
	e_sector1	provincia	text	
	e_sector2	pais	text	
	agr_parcela	hectareas_parc	double	
	agr_tipo_suelo	suelo	text	
	agr_cultivo	cultivo	text	
	agr_cultivo	hectareas_cult	double	
	agr_tipo_riego	riego	text	

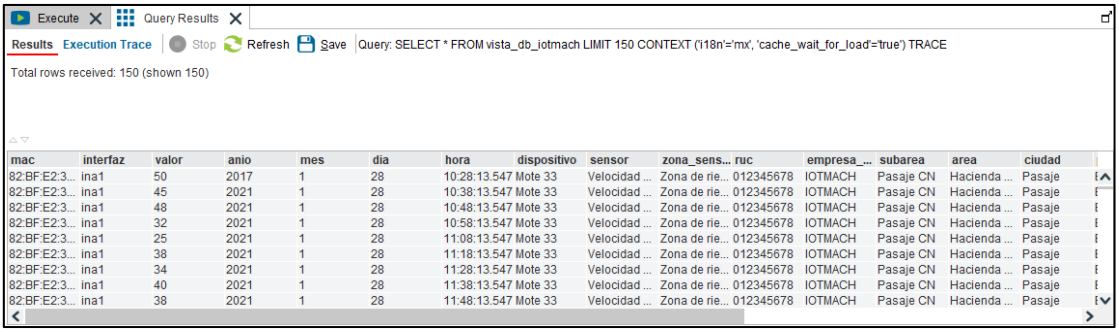
Fuente: Denodo

Finalmente se guarda la vista, para ver la información de la vista, se selecciona la misma y se escoge la opción *Execution Panel* que se encuentra ubicado en la parte superior de la pestaña. Se despliega una pestaña desde la parte inferior y se da clic sobre el botón *Execute*.

Gráfico 40: Proceso para ejecutar vista integrada

Fuente: Denodo

Gráfico 41: Resultado de ejecutar vista integrada



Fuente: Denodo

2.5.2.3. Almacenamiento distribuido

El almacenamiento distribuido implica el uso de un clúster con Hadoop, la información sobre el proceso para la implementación de un clúster en máquinas virtuales se encuentra en la documentación del proyecto [79]. La arquitectura utilizada para este proyecto es la siguiente:

Tabla 4: Recursos asignados en arquitectura clúster

Recurso	Nodo 1	Nodo 2	Nodo 3
Tipo	Master	Esclavo	Esclavo
Memoria RAM	12 GB	4 GB	4 GB
Almacenamiento	100 GB	100 GB	100 GB
Hilos de procesamiento	3	1	1

Fuente: Elaboración propia

Todos los nodos se comunican por medio de una red LAN distribuida de la siguiente manera:

Tabla 5: Direcciones IP red LAN clúster

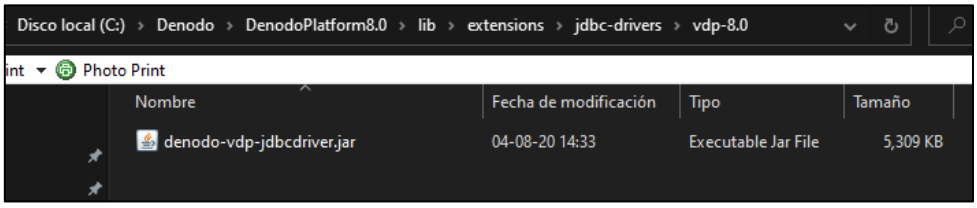
Nodo	IP
Nodo 1 – Master	192.168.10.100/24
Nodo 2 – Esclavo	192.168.10.101/24
Nodo 3 – Esclavo	192.168.10.102/24

Fuente: Elaboración propia

La contraseña de acceso para todos los nodos es *master*.

Una vez dentro del clúster es necesario tener el driver que permita la conexión con Denodo, mismo que se encuentra en el directorio de instalación lib/extensions/jdbc-drivers/vdp-8.0.

Gráfico 42: Directorio driver de conexión a Denodo vía JDBC

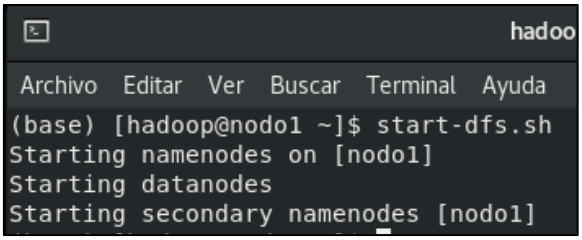


Fuente: Elaboración propia

Si el driver está alojado en la máquina anfitrión, debe buscar la manera para alojarlo en la máquina virtual del nodo master, para este proyecto se utilizó una transferencia por medio de la nube.

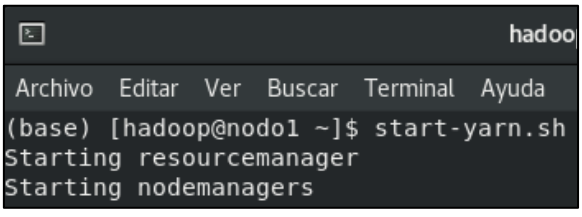
Para ejecutar el clúster debe tener todas las máquinas virtuales encendidas, y comprobar que todas tengan conexión entre sí, una vez hecha esa comprobación, debe levantar los servicios de HDFS y YARN, para ello debe emplear los siguientes comandos.

Gráfico 43: Comando para iniciar el servicio HDFS



Fuente: Clúster Hadoop

Gráfico 44: Comando para iniciar el servicio YARN



Fuente: Clúster Hadoop

Lo siguiente es ejecutar un script en Python para extraer los datos virtualizados desde Denodo, para una mayor facilidad de trabajo y ver resultados paso a paso, se utiliza la herramienta Jupyter-Notebook.

Algoritmo 7: Conexión a Denodo desde Python

```
denodoserver_name = "192.168.1.8"
denodoserver_jdbc_port = "9999"
denodoserver_database = "db_iotmach"
denodoserver_uid = "admin"
denodoserver_pwd = "admin"
denododriver_path = "/home/hadoop/CuadernosJupyter/denodo-vdp-
jdbcdriver.jar"

conn_uri = "jdbc:vdb://%s:%s/%s" % (denodoserver_name,
denodoserver_jdbc_port, denodoserver_database)

cnxn = dbdriver.connect( "com.denodo.vdp.jdbc.Driver", conn_uri,
driver_args = {"user": denodoserver_uid, "password":
denodoserver_pwd}, jars = denododriver_path )
```

Fuente: Elaboración propia

Es necesario establecer los parámetros de conexión como el hostname, puerto, base de datos, usuario, contraseña y el directorio en donde se encuentra el driver de conexión anteriormente buscado. Luego se arma una URL de conexión y finalmente se realiza la conexión como tal por medio del comando dbdriver.connect.

Algoritmo 8: Consulta de la vista integrada

```
query = "select * from vista_db_iotmach CONTEXT ('QUERYTIMEOUT' =
'3600000')"

cur = cnxn.cursor()
cur.execute(query)

results = cur.fetchall()

cabeceras = []
for cabecera in cur.description:
    cabeceras.append(cabecera[0])
```

Fuente: Elaboración propia

Se almacena la consulta de la vista integrada en una variable para luego ser ejecutada mediante el comando execute, posteriormente el resultado de la consulta se almacena en una variable y antes de ser almacenado en HDFS hay que recordar que este tipo de consulta no retorna una cabecera dentro del conjunto de datos obtenido, para ello en una variable se almacenan las cabeceras haciendo un recorrido al objeto *description* del resultado de la consulta.

Algoritmo 9: Almacenar resultados de la consulta en archivo CSV

```
df = pd.DataFrame(results)
df.columns = cabeceras
df.to_csv (r'/home/hadoop/CuadernosJupyter/vista_iotmach.csv', index
= False)
```

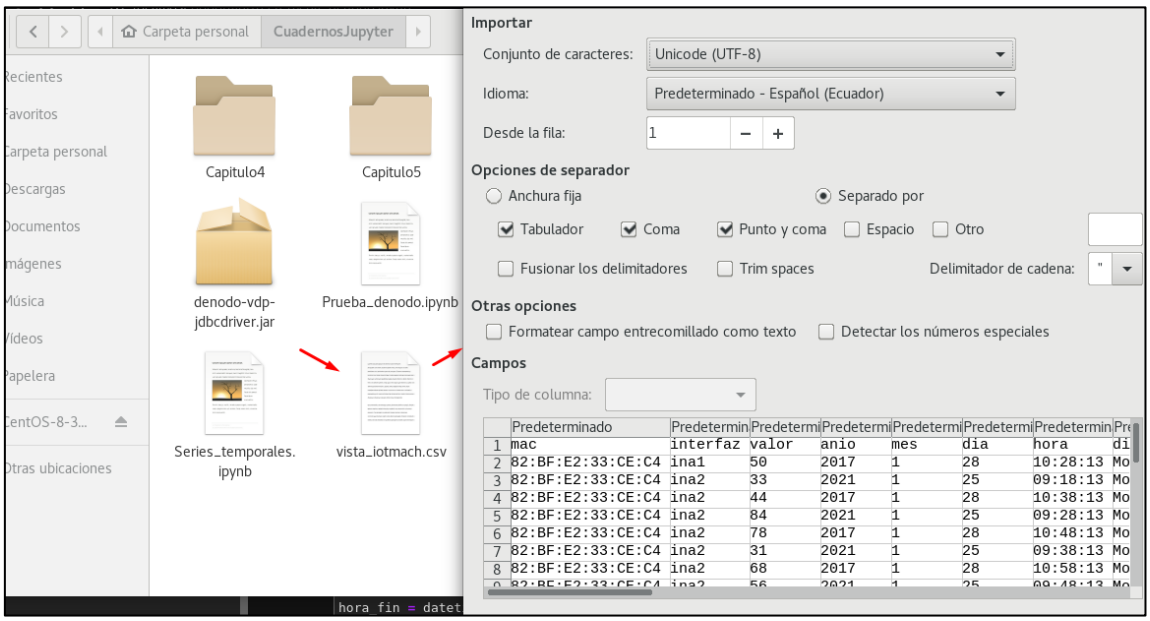
Fuente: Elaboración propia

Para almacenar datos en HDFS, es necesario que estos se encuentren alojados en un fichero, es por eso, que los resultados de la consulta se transforman en un

dataframe gracias a la librería pandas, a ese dataframe se le adjunta las cabeceras y se los guarda como archivo csv.

Como se puede apreciar en el **Gráfico 45**, los datos fueron almacenados en un fichero CSV dentro del directorio establecido.

Gráfico 45: Archivo CSV de vista integrada de Denodo



Fuente: Elaboración propia

Algoritmo 10: Almacenamiento de datos en HDFS

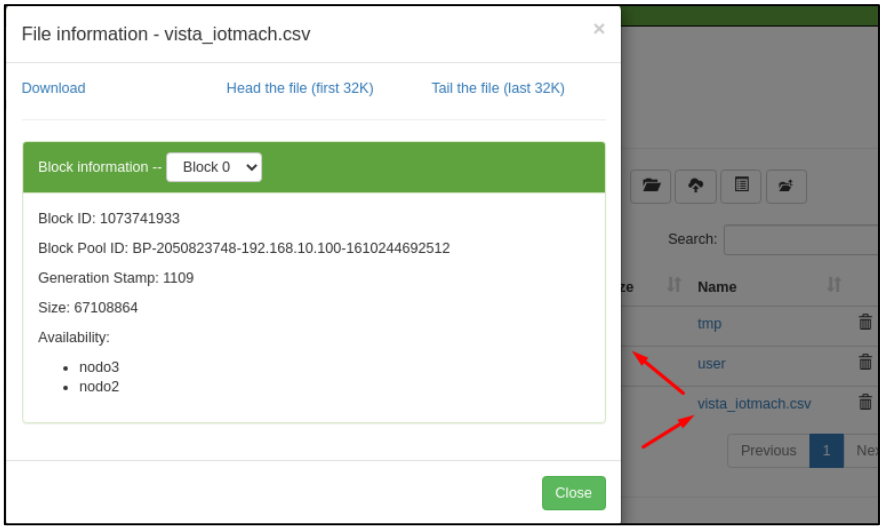
```
hdfs=HDFileSystem(host='nodo1',port=9000)

hdfs.put('/home/hadoop/CuadernosJupyter/vista_iotmach.csv',
'/vista_iotmach.csv')
```

Fuente: Elaboración propia

Una vez que los datos se han almacenado en un fichero CSV, se procede a realizar la conexión a HDFS y mediante el comando *put*, almacenar los datos que se encuentran en el archivo CSV a HDFS.

Como se aprecia en el **Gráfico 46**, los datos no solo se encuentran almacenados en un solo nodo, si no, que está replicado en todos los nodos que conforman el clúster, esta información se puede ver a través del navegador en la dirección nodo:9870, pestaña Utilities > Browse the file system, ubicar el archivo, dar clic sobre él y se despliega la información de los nodos en donde está replicado.

Gráfico 46: Información de archivo almacenado en HDFS



Fuente: Clúster Hadoop

2.5.3. Aplicación de técnicas de análisis de datos

Los lenguajes de programación Python y R poseen una gran cantidad de librerías para aplicar técnicas de análisis, las técnicas que se muestran a continuación son aplicadas sobre datos de sensores generados durante 4 años (2017-2021).

2.5.3.1. Carga y filtro de datos

Antes de aplicar los modelos estadísticos predictivos, es necesario preparar los datos, es decir, cargarlos desde HDFS y realizar filtros, los datos que se generaron fueron datos de lecturas con una frecuencia de 10 minutos y para su análisis, se realiza un promedio mensual de las lecturas por los 4 años.

Algoritmo 11: Carga y preparación de datos

```
sc = SparkContext(appName='ser_tem_iotmach', master = 'yarn')
spark=SparkSession.builder.getOrCreate()

ruta = "/vista_iotmach.csv"
file_type = "csv"
infer_schema = "false"
first_row_is_header = "true"
delimiter = ","
dataset = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(ruta)

dataset_f = dataset.filter((dataset.mac == '82:BF:E2:33:CE:C4') & \
               (dataset.sensor == 'Temperatura relativa del aire'))

df=dataset_f.toPandas()

df.fecha = pd.to_datetime(df.fecha)
df = df.sort_values(by=['fecha'])
```

Fuente: Elaboración propia

La carga y preparación consiste en crear una sesión en Spark a través del servicio YARN para la ejecución de procesos en paralelo dentro del clúster, una vez creada la sesión, se procede a configurar los parámetros para cargar los datos de los sensores almacenados en HDFS, al cargar los datos con el comando *spark.read.format*, estos no se presentan en forma de dataframe, si no, con un formato spark, utilizando comandos spark, se hace el filtro por Mac y sensor, para este proyecto, la muestra consiste en datos correspondientes al dispositivo *82:BF:E2:33:CE:C4* y al sensor de *Temperatura relativa del aire*, en el **Anexo 1**, se podrá observar la aplicación de las mismas técnicas de análisis para distintos sensores y dispositivos. Una vez realizado el filtro se convierten los datos a una estructura de tipo dataframe, los campos fecha a tipo datetime y se los ordena de forma descendente.

Gráfico 47: Datos de sensores cargados desde HDFS

	mac	interfaz	valor	anio	mes	dia	hora	dispositivo	sensor	zona_sensor	...	area	ciudad	provincia	pais	hectareas_parc
135416	82:BF:E2:33:CE:C4	ina3	31	2017	1	28	10:38:13	Mote 33	Temperatura relativa del aire	Zona de riego Norte	...	Hacienda Pasaje	Pasaje	El Oro	Ecuador	200.0
135418	82:BF:E2:33:CE:C4	ina3	30	2017	1	28	10:48:13	Mote 33	Temperatura relativa del aire	Zona de riego Norte	...	Hacienda Pasaje	Pasaje	El Oro	Ecuador	200.0
135420	82:BF:E2:33:CE:C4	ina3	27	2017	1	28	10:58:13	Mote 33	Temperatura relativa del aire	Zona de riego Norte	...	Hacienda Pasaje	Pasaje	El Oro	Ecuador	200.0
135422	82:BF:E2:33:CE:C4	ina3	34	2017	1	28	11:08:13	Mote 33	Temperatura relativa del aire	Zona de riego Norte	...	Hacienda Pasaje	Pasaje	El Oro	Ecuador	200.0
135424	82:BF:E2:33:CE:C4	ina3	31	2017	1	28	11:18:13	Mote 33	Temperatura relativa del aire	Zona de riego Norte	...	Hacienda Pasaje	Pasaje	El Oro	Ecuador	200.0

Fuente: Elaboración propia

Algoritmo 12: Promedio mensual de las lecturas del sensor de temperatura del aire

```
df.index = df.fecha

df.valor = df.valor.astype('int')
ts_prom_mes = df.valor.resample('M').mean()
```

Fuente: Elaboración propia

Para crear una serie temporal, se reindexa el dataframe tomando los valores de la fecha, se procede a cambiar el tipo de dato de los valores del campo valor y se realiza el promedio mensual por medio del comando *resample().mean()*.

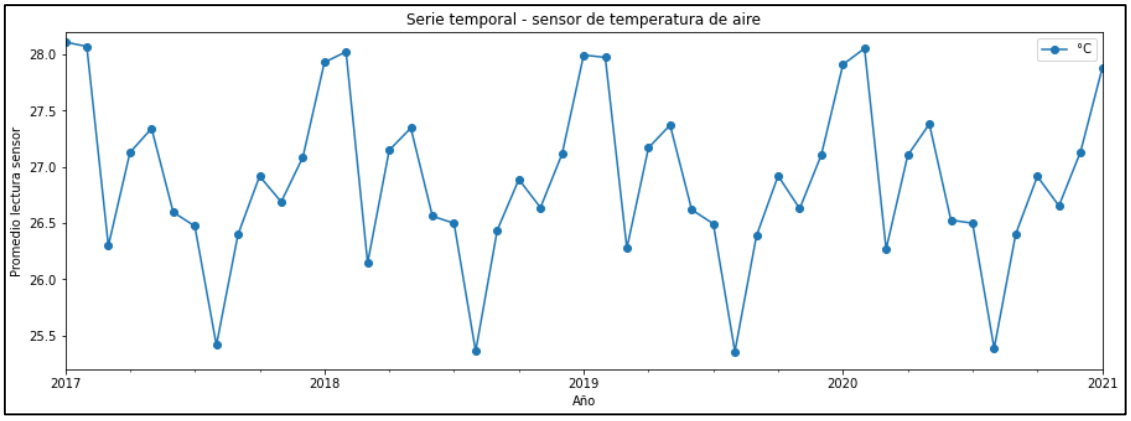
Algoritmo 13: Gráfico serie temporal

```
fig, ax = plt.subplots()
fig.set_size_inches(15, 5)
ts_prom_mes.plot(linestyle='-', marker = 'o')
ax.legend(['°C'], loc=1)
ax.set_title('Serie temporal - sensor de temperatura de aire')
ax.set_xlabel("Año")
ax.set_ylabel("Promedio lectura sensor")
ax.set_ylim([25.2,28.2])
```

Fuente: Elaboración propia

Con la ayuda de los comandos de la librería matplotlib, es posible generar un gráfico estadístico de la serie temporal como se muestra en el **Gráfico 48**.

Gráfico 48: Gráfico serie temporal de sensor de temperatura del aire



Fuente: Elaboración propia

2.5.3.2. Estadística Básica

En el **Algoritmo 14**, se observan los comandos en Python utilizados para calcular la media, mediana, moda, varianza, desviación estándar y coeficiente de variación.

Algoritmo 14: Estadística Básica

```
media = round(df_prom_mes.mean(), 2).values[0]
mediana = round(df_prom_mes.median(), 2).values[0]
moda = round(df_prom_mes.mode(), 2).values[0][0]
varianza = round(df_prom_mes.var(), 2).values[0]
desv_est = round(df_prom_mes.std(ddof=0), 2).values[0]
coef_var = round(desv_est/media, 2)
```

Fuente: Elaboración propia

Tabla 6: Resultados estadística básica

Media	26.88
Mediana	26.89
Moda	25.36
Varianza	0.53
Desviación estándar	0.72
Coeficiente de variación	0.03

Fuente: Elaboración propia

Los resultados presentados en la **Tabla 7**, para el promedio mensual desde el año 2017 hasta el 2021 del sensor de temperatura relativa del aire, indican que, el valor promedio de temperatura del aire es de 26.88, el valor que se encuentra en la mitad del conjunto de datos es 26.89, el valor que más se repite es 25.36, los datos se encuentran próximos a media ya que su varianza es de 0.53 y su desviación estándar es de 0.72.

2.5.3.3. Estadística Descriptiva

Asimetría y curtosis

Algoritmo 15: Asimetría y curtosis

```
asimetria = round(skew(promedio_mes, bias=False), 2)
curtosis = round(kurtosis(promedio_mes, bias=False), 2)
print(f'Asimetria: {asimetria}\nCurtosis: {curtosis}')
```

```
Asimetria: -0.12
Curtosis: -0.17
```

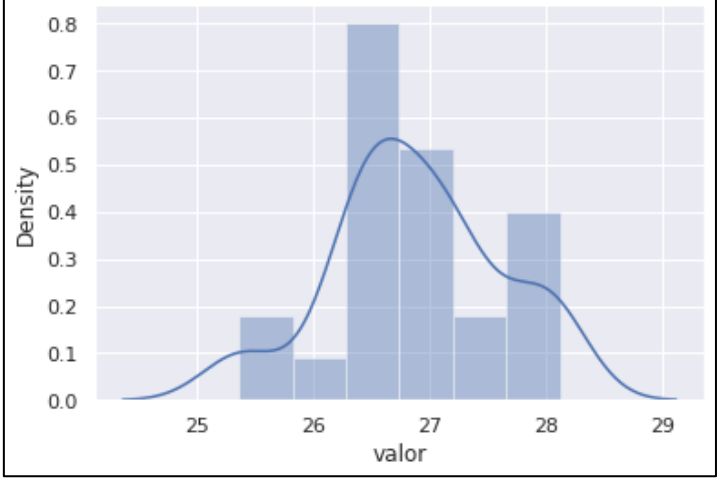
Fuente: Elaboración propia

El resultado de asimetría menor a cero, indica que la cola de distribución de los valores respecto a la media se desplaza ligeramente hacia la izquierda, mientras que su curtosis, al ser menos que cero indica que los datos se encuentran muy cerca de la media.

Histograma y diagrama de densidad

Gráfico 49: Histograma y Diagrama de densidad

```
sns.set(color_codes=True)
sns.distplot(promedio_mes)
```



Fuente: Elaboración propia

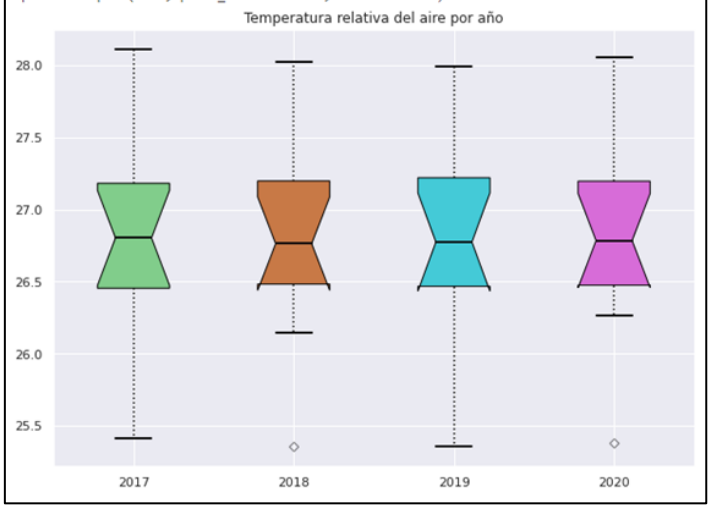
Como se puede apreciar, los datos se centran entre los valores 26 y 27, su distribución está ligeramente hacia la izquierda respecto a la media, afirmando los resultados de otros estadísticos como la asimetría y curtosis, calculados anteriormente.

Diagrama de cajas

Para elaborar los diagramas de cajas, se han filtrado el mismo conjunto de datos por años.

Gráfico 50: Diagrama de cajas

```
promedio_mes_2017 = df_c2017.valor.resample('M').mean()
df_prom_mes_2017 = pd.DataFrame(promedio_mes_2017.values)

promedio_mes_2018 = df_c2018.valor.resample('M').mean()
df_prom_mes_2018 = pd.DataFrame(promedio_mes_2018.values)

promedio_mes_2019 = df_c2019.valor.resample('M').mean()
df_prom_mes_2019 = pd.DataFrame(promedio_mes_2019.values)

promedio_mes_2020 = df_c2020.valor.resample('M').mean()
df_prom_mes_2020 = pd.DataFrame(promedio_mes_2020.values)

data = [df_prom_mes_2017[0], df_prom_mes_2018[0], df_prom_mes_2019[0], df_prom_mes_2020[0]]

fig = plt.figure(figsize =(10, 7))
ax = fig.add_subplot(111)
bp = ax.boxplot(data, patch_artist = True, notch ='True')
```



Fuente: Elaboración propia

Al interpretar el gráfico, se observa que el valor medio de cada conjunto de datos por año es bastante similar, al igual que sus extremos superiores, la diferencia más notoria radica en los años 2018 y 2020, en donde sus límites inferiores son más cercanos a su valor medio y presentan valores atípicos

2.5.3.4. Estadística predictiva

ARIMA

ARIMA (p, d, q), Lo primero es determinar si la serie temporal actual es estacionaria a través de la prueba de Dickey-Fuller.

Algoritmo 16: Prueba de Dickey-Fuller

```
result=adfuller(df)
labels = ['ADF Test Statistic','p-value','#Lags Used','Number of
Observations']
for value,label in zip(result,labels):
    print(label+' : '+str(value) )

if result[1] <= 0.05:
    print("Es estacionaria")
else:
    print("No es estacionaria")
```

Fuente: Elaboración propia

Al ejecutar la prueba de Dickey-Fuller, el valor de la variable p debe estar por debajo de 0.05, para el conjunto de datos actual, los resultados indican que no es una serie temporal estacionaria.

Gráfico 51: Resultado prueba Dickey-Fuller

```
ADF Test Statistic : -2.824927150350611
p-value : 0.05480037625896768
#Lags Used : 11
Number of Observations : 37
No es estacionaria
```

Fuente: Elaboración propia

Existen varias formas de convertir una serie temporal a estacionaria, la más utilizada es la aplicación de diferencias. Para la serie temporal utilizada, bastó con realizar una diferencia por medio del comando *diff()*.

Al aplicar nuevamente la prueba de Dickey-Fuller, se puede observar el cambio de la variable p a un valor menor a 0.05.

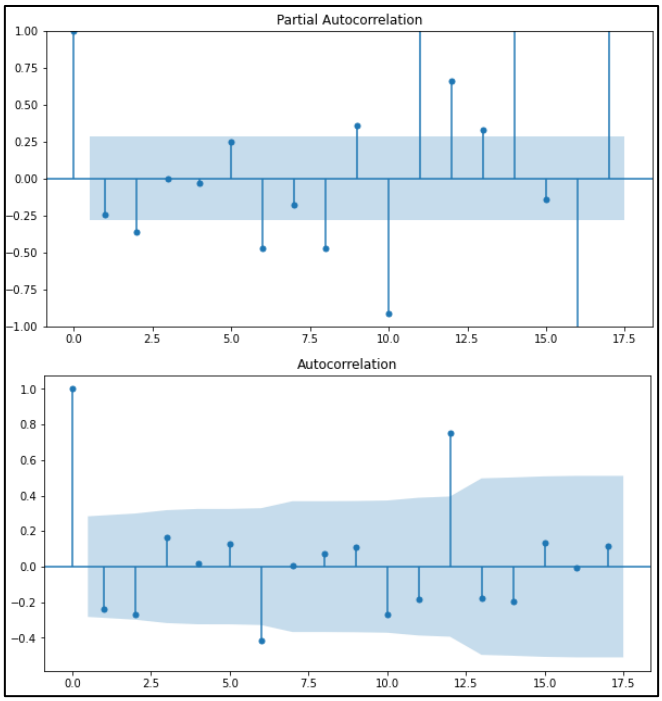
Gráfico 52: Prueba Dickey-Fuller de la serie temporal con una diferencia

```
ADF Test Statistic : -16.642137729630186
p-value : 1.6311771948998584e-29
#Lags Used : 10
Number of Observations : 37
Es estacionaria
```

Fuente: Elaboración propia

El siguiente paso es analizar los gráficos de autocorrelación y autocorrelación parcial aplicando los comandos plot_acf() y plot_pacf() respectivamente. El análisis de los gráficos consiste en identificar aquellos indicadores que están por encima del límite determinado por el área sombreada. Conforme a la información brindada por los gráficos se puede determinar que el valor del parámetro p están entre 4 y 6, mientras que el valor de q, ronda los valores entre 0 y 2.

Gráfico 53: Gráfico de autocorrelación y autocorrelación parcial



Fuente: Elaboración propia

Luego de varias pruebas, el modelo más óptimo se obtuvo con parámetros p, d, q de 6, 1 y 1 respectivamente.

Algoritmo 17: Prueba de Dickey-Fuller

```
model = ARIMA(ts_prom_mes, order=(6,1,1))
model_fit = model.fit()
print(model_fit.summary())
```

Fuente: Elaboración propia

Gráfico 54: Resumen métricas de modelo ARIMA

```
                               SARIMAX Results                                
==============================================================================
Dep. Variable:                  valor   No. Observations:                   49
Model:                 ARIMA(6, 1, 1)   Log Likelihood                -37.533
Date:                Mon, 15 Mar 2021   AIC                             91.065
Time:                        10:20:49   BIC                            106.035
Sample:                  01-31-2017   HQIC                            96.722
                       - 01-31-2021                                         
Covariance Type:                opg                                         
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.2325      0.166      1.398      0.162      -0.093       0.558
ar.L2         -0.2838      0.236     -1.204      0.229      -0.746       0.178
ar.L3          0.2291      0.275      0.833      0.405      -0.310       0.768
ar.L4         -0.2363      0.275     -0.859      0.390      -0.775       0.303
ar.L5         -0.0031      0.153     -0.020      0.984      -0.303       0.296
ar.L6         -0.6469      0.171     -3.773      0.000      -0.983      -0.311
ma.L1         -0.9994     26.970     -0.037      0.970     -53.859      51.860
sigma2         0.2326      6.266      0.037      0.970     -12.048      12.513
===================================================================================
Ljung-Box (L1) (Q):                   0.58   Jarque-Bera (JB):                 3.48
Prob(Q):                              0.45   Prob(JB):                         0.18
Heteroskedasticity (H):               0.68   Skew:                            -0.62
Prob(H) (two-sided):                  0.45   Kurtosis:                         2.57
===================================================================================
```

Fuente: Elaboración propia

Como se observa en el **Gráfico 54**, el resumen de métricas al modelo entrenado indica su eficiencia, por ejemplo, su valor de p se encuentra por debajo del nivel de significancia 0.05 y el valor de AIC a comparación de otras pruebas realizadas con otros valores a los parámetros p, d, q es menor.

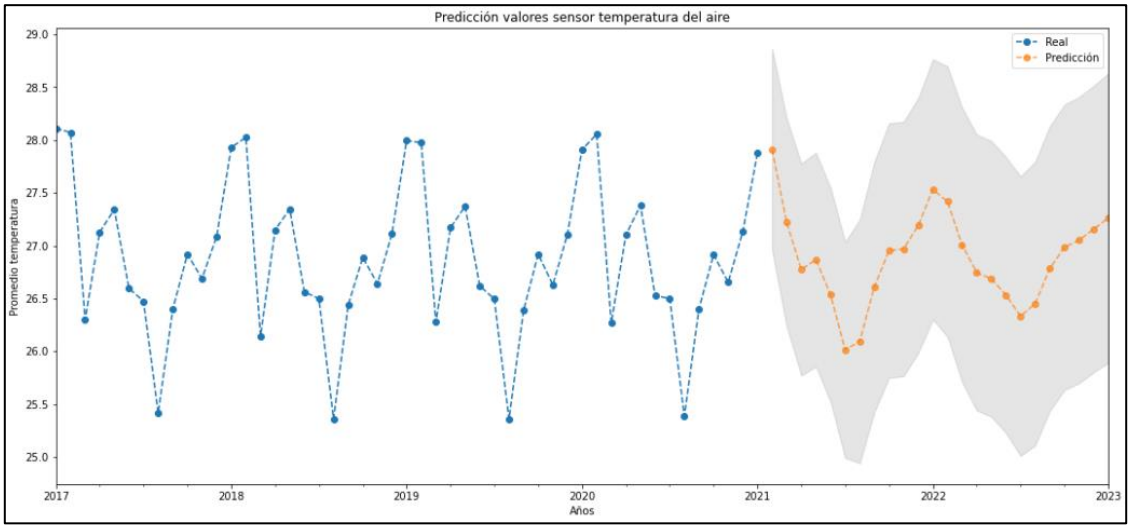
Algoritmo 18: Predicción de modelo ARIMA

```
pred = model_fit.get_forecast(steps=24)

pred_ci = pred.conf_int()
ax = ts_prom_mes.plot(label='Real', linestyle="--", marker = 'o')
pred.predicted_mean.plot(ax=ax, label='Predicción', alpha=.7,
figsize=(20, 7), linestyle="--", marker = 'o')
ax.fill_between(pred_ci.index,
                pred_ci.iloc[:, 0],
                pred_ci.iloc[:, 1], color='k', alpha=.1)
ax.set_xlabel('Años')
ax.set_ylabel('Promedio temperatura')
ax.set_title('Predicción valores sensor temperatura del aire')
plt.legend()
plt.show()
```

Fuente: Elaboración propia

Para realizar una predicción se utiliza el comando get_forecast(), al cual se le envía como parámetros la cantidad de pasos a predecir, al tratarse de una serie temporal de promedio mensuales, la cantidad de pasos a predecir en este ejemplo son 24, es decir, dos años.

Gráfico 55: Predicción del modelo ARIMA



Fuente: Elaboración propia

Como se puede apreciar en el **Gráfico 55**, la predicción de acuerdo al modelo, indica que los valores de temperatura, conforme avanzan los años, pierden similitud a los datos reales.

SARIMA

El proceso para obtener los parámetros p, d, q es el mismo, y considerando que se trata del mismo conjunto de datos, no se lo volverá a detallar, para los parámetros P, D, Q y m, es necesario analizar la serie temporal y determinar en qué punto la tendencia estacionaria comienza a repetirse, en el conjunto de datos utilizado, le tendencia comienza a repetirse al pasar los doce meses, es por ello, que el periodo estacional (m) toma el valor de 12.

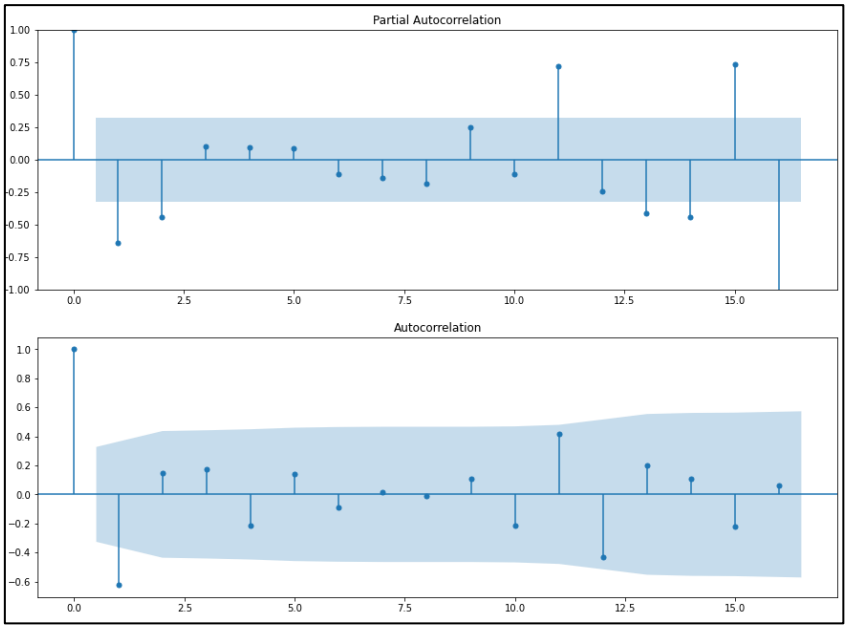
El siguiente paso es determinar el número de diferencias aplicando el periodo estacional y a partir de eso, analizar las gráficas de autocorrelación y autocorrelación parcial.

Algoritmo 19: Diferencia, autocorrelación y autocorrelación parcial estacional

```
diff_est_sarima = diff1_sarima.diff(periods=12).dropna()

plot_pacf(diff_est_sarima)
plt.ylim(-1, 1)
plt.show()

plot_acf(diff_est_sarima)
plt.show()
```

Fuente: Elaboración propia

Gráfico 56: Autocorrelación y autocorrelación parcial estacional



Fuente: Elaboración propia

El análisis de los gráficos de autocorrelación parcial y autocorrelación, indican que los valores para P y Q son de 2 y 0 respectivamente, por lo tanto, ya se tienen los valores necesarios para crear y entrenar el modelo SARIMA.

Algoritmo 20: Creación y entrenamiento del modelo SARIMA

```
modelo_sarima = sm.tsa.statespace.SARIMAX(df_sarima_prom_mes,
order=(6,1,1), seasonal_order=(2,1,0,12),
                                          enforce_stationarity=False,
enforce_invertibility=False)
sarima_fit = modelo_sarima.fit()
print(sarima_fit.summary())
```

Fuente: Elaboración propia

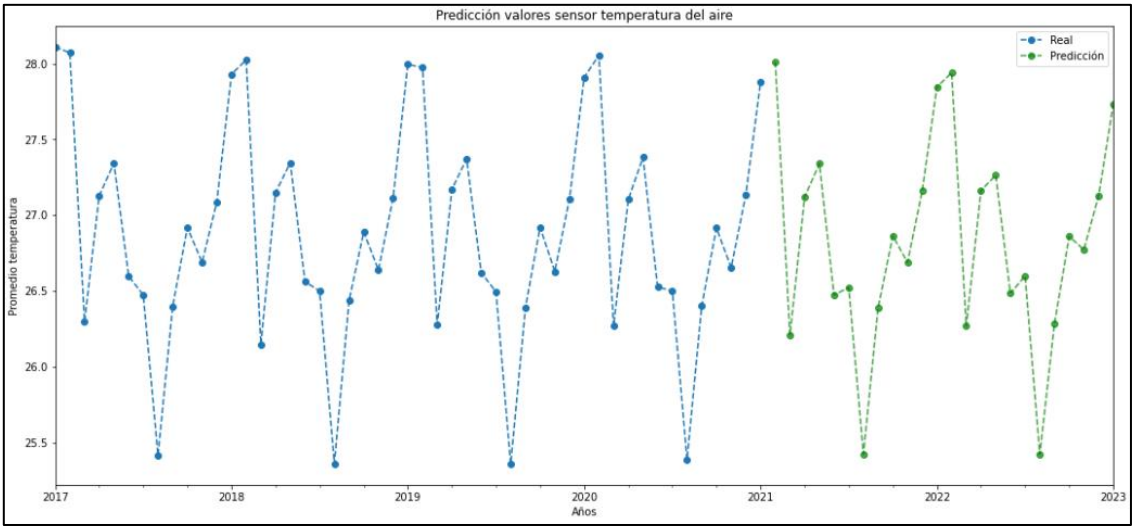
Gráfico 57: Resumen métricas de modelo ARIMA

```
==========================================================================================
Dep. Variable:                               valor   No. Observations:                   49
Model:            SARIMAX(6, 1, 1)x(2, 1, [], 12)   Log Likelihood                41.819
Date:                            Mon, 15 Mar 2021   AIC                           -63.638
Time:                                    15:26:36   BIC                           -65.721
Sample:                                01-31-2017   HQIC                          -71.974
                                     - 01-31-2021
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.0671      0.002     34.345      0.000       0.063       0.071
ar.L2         -0.4648      0.003   -180.133      0.000      -0.470      -0.460
ar.L3          0.0302      0.003      9.420      0.000       0.024       0.036
ar.L4          0.1822      0.002     82.954      0.000       0.178       0.186
ar.L5         -0.3818      0.001   -355.440      0.000      -0.384      -0.380
ar.L6         -0.5249      0.002   -341.439      0.000      -0.528      -0.522
ma.L1         -0.7480   4.02e-06  -1.86e+05      0.000      -0.748      -0.748
ar.S.L12      -0.1320      0.003    -47.958      0.000      -0.137      -0.127
ar.S.L24      -0.3591      0.002   -231.834      0.000      -0.362      -0.356
sigma2      5.43e-08   9.11e-08      0.596      0.551   -1.24e-07    2.33e-07
==========================================================================================
Ljung-Box (L1) (Q):                   0.43   Jarque-Bera (JB):                 0.93
Prob(Q):                              0.51   Prob(JB):                         0.63
Heteroskedasticity (H):               0.40   Skew:                             0.92
Prob(H) (two-sided):                  0.57   Kurtosis:                         2.46
==========================================================================================
```

Fuente: Elaboración propia

Al igual que en el modelo ARIMA, las métricas de resumen son similares, entre las más importantes, se encuentra su valor p, el cual también está por debajo del nivel de significancia 0.05 y el valor de AIC es mucho menor que el de ARIMA.

Gráfico 58: Predicción modelo SARIMA



Fuente: Elaboración propia

Los mismos comandos que en el modelo ARIMA se aplican para realizar la predicción en el modelo SARIMA, a diferencia del modelo ARIMA, visualmente los datos de predicción tienen un comportamiento bastante similar a los datos reales.

Holt-Winters

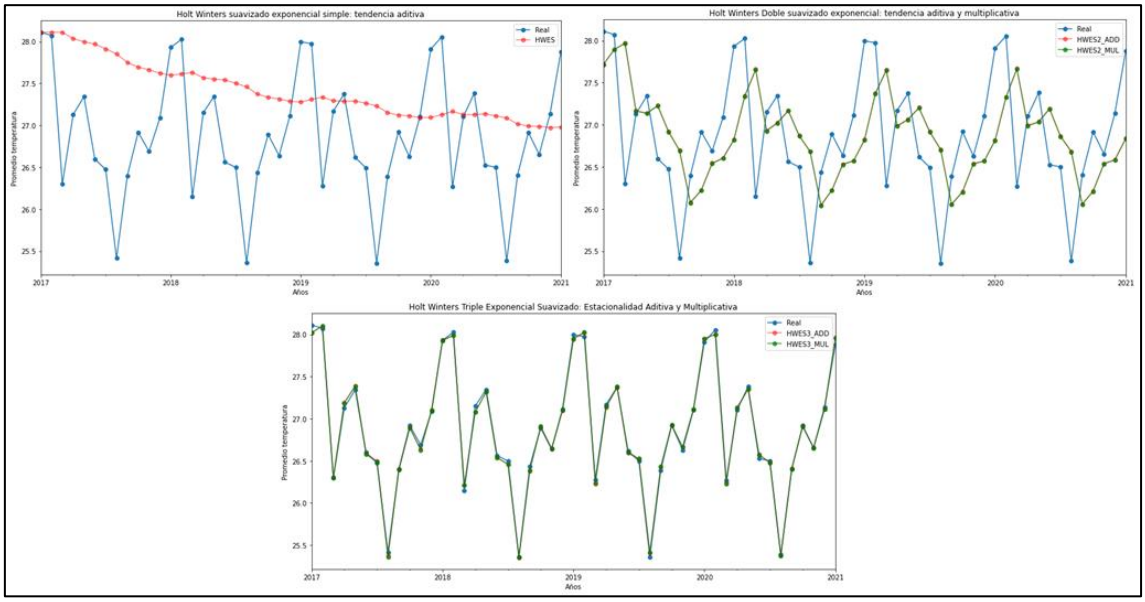
El modelo Holt-Winters es una variante del suavizado exponencial, al actual conjunto de datos se le aplicó un suavizado exponencial aditivo y multiplicativo, simple, doble y triple a través de los comandos presentados en la **Algoritmo 21**.

Algoritmo 21: Suavizado exponencial aditivo y multiplicativo, simple, doble y triple

```
hwes =
SimpleExpSmoothing(dfhw_prom_mes).fit(smoothing_level=alpha,optimize
d=False,use_brute=True).fittedvalues

HWES2_ADD =
ExponentialSmoothing(dfhw_prom_mes,trend='add').fit().fittedvalues
HWES2_MUL =
ExponentialSmoothing(dfhw_prom_mes,trend='mul').fit().fittedvalues

HWES3_ADD =
ExponentialSmoothing(dfhw_prom_mes,trend='add',seasonal='add',season
al_periods=12).fit().fittedvalues
HWES3_MUL =
ExponentialSmoothing(dfhw_prom_mes,trend='mul',seasonal='mul',season
al_periods=12).fit().fittedvalues
```

Fuente: Elaboración propia

Gráfico 59: Suavizado exponencial aditivo y multiplicativo, simple, doble y triple



Fuente: Elaboración propia

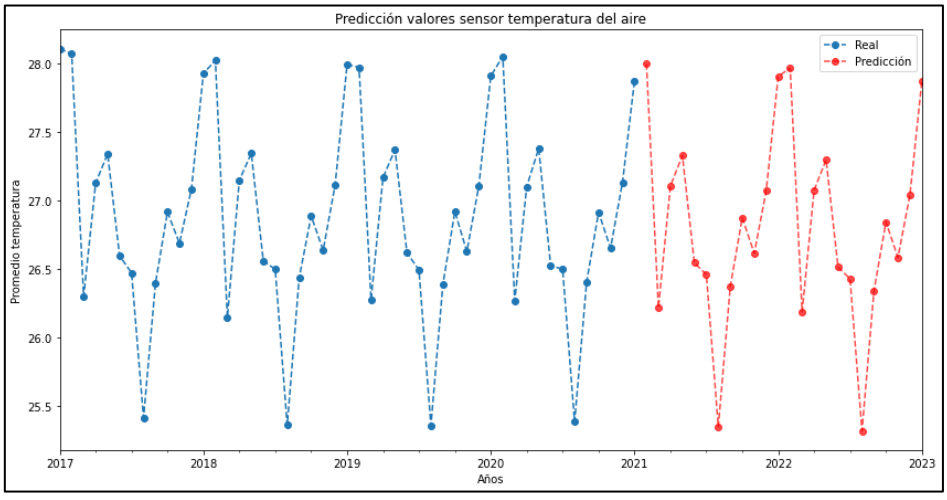
Como se puede apreciar en el **Gráfico 59**, el suavizado exponencial triple aditivo y multiplicativo es el que más se ajusta a la tendencia real de los datos por lo cual es el más óptimo para realizar la predicción de los datos.

Algoritmo 22: Predicción modelo Holt-Winters

```
fitted_model =
ExponentialSmoothing(dfhw_prom_mes,trend='mul',seasonal='mul',season
al_periods=12).fit()
predictions = fitted_model.forecast(24)

ax = dfhw_prom_mes.plot(label='Real', linestyle="--", marker="o")
predictions.plot(ax=ax, label='Predicción', alpha=.7, figsize=(14,
7), color="red", linestyle="--", marker="o")
ax.set_xlabel('Años')
ax.set_ylabel('Promedio temperatura')
ax.set_title('Predicción valores sensor temperatura del aire')
plt.legend()
plt.show()
```

Fuente: Elaboración propia

Gráfico 60: Predicción modelo Holt-Winters



Fuente: Elaboración propia

De acuerdo al **Gráfico 60**, y a comparación con los modelos ARIMA y SARIMA, la predicción realizada por el modelo Holt-Winters aplicando el suavizado exponencial triple con tendencia aditiva y multiplicativa es bastante similar a la predicción realizada por el modelo SARIMA, es decir, sus predicciones siguen la tendencia de los datos reales.

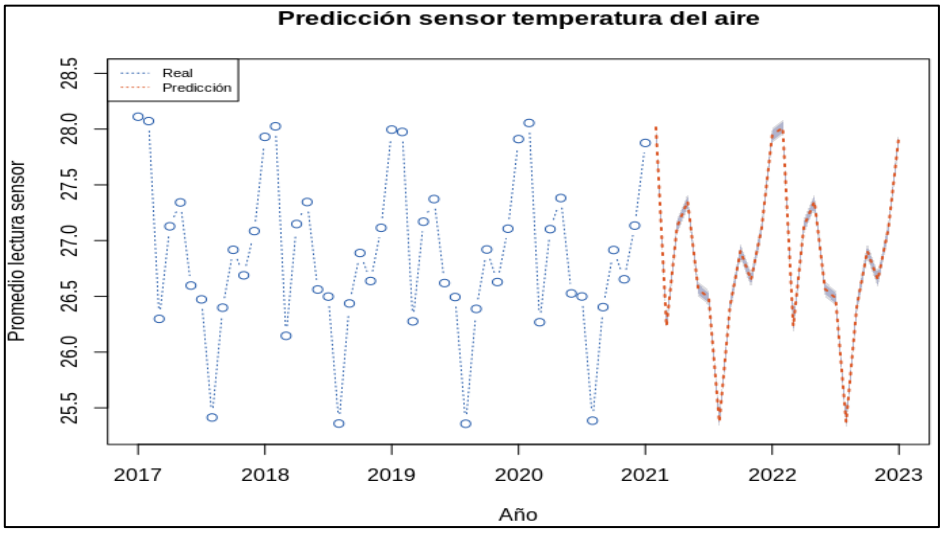
STLM

Algoritmo 23: Creación y entrenamiento modelo STLM

```
stlm_model <- stlm(ts_stl)
pred_stlm<- forecast(stlm_model, h=24)

plot(pred_stlm, col="#265CAB" , fcol= "#DC5624", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", ylim=c(25.3,28.5))

legend("topleft", legend=c("Real", "Predicción"),
       col=c("#265CAB", "#DC5624"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 61: Predicción modelo STLM



Fuente: Elaboración propia

Los resultados de la predicción indican que los datos predichos se ajustan a los datos reales y se asemejan a los resultados de los modelos Holt-Winters y SARIMA, visualmente.

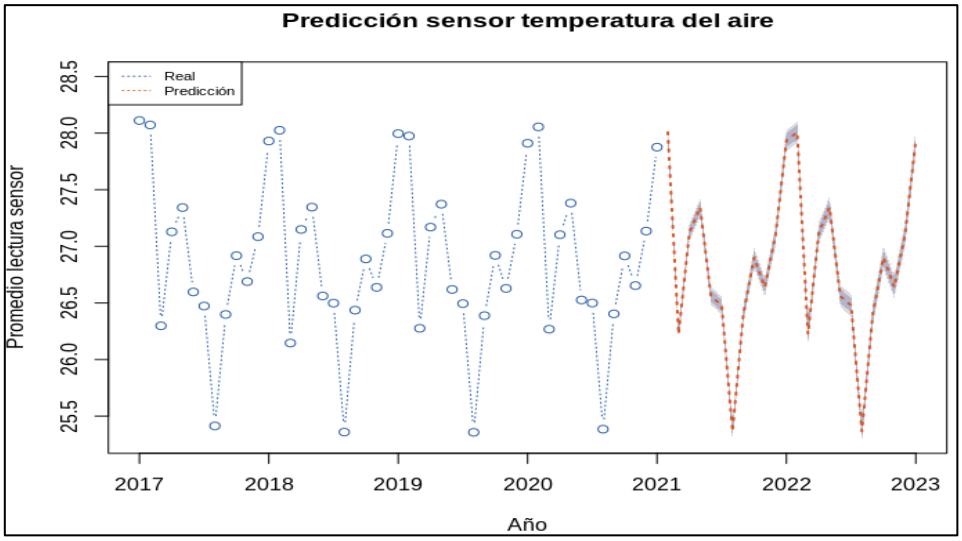
STLF

Algoritmo 24: Creación y entrenamiento modelo STLF

```
stlf_model <- stlf(ts_stl, lambda = 0)
pred_stlf<- forecast(stlf_model, h=24)

plot(pred_stlf, col="#265CAB" , fcol= "#DC5624", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", ylim=c(25.3,28.5))

legend("topleft", legend=c("Real", "Predicción"),
       col=c("#265CAB", "#DC5624"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 62: Predicción modelo STLF



Fuente: Elaboración propia

A comparación con los modelos Holt-Winters, SARIMA y STLM, su predicción presenta gran similitud a los datos reales y a las predicciones de los modelos mencionados.

TBATS

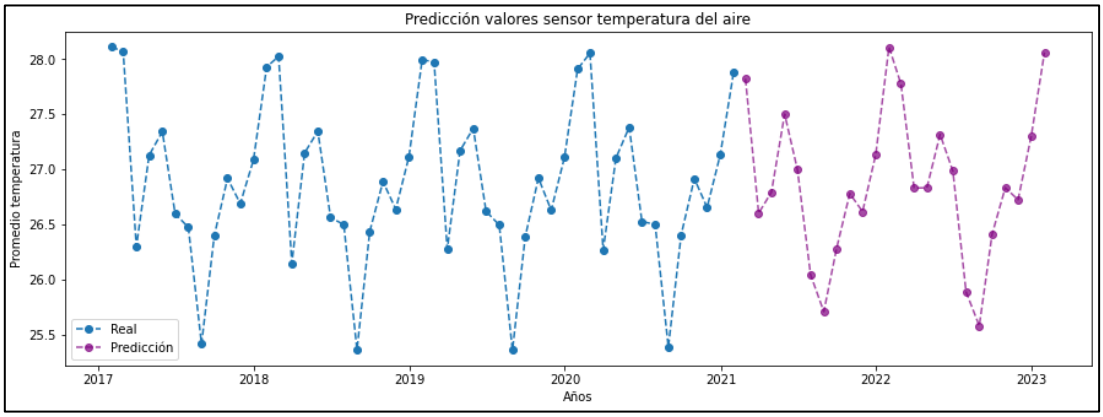
Al igual que Holt-Winters, TBATS también es un modelo basado en suavizado exponencial y su gran ventaja es el ajuste automático que el modelo posee para adaptarse a cualquier tipo de serie temporal.

Algoritmo 25: Entrenamiento y predicción del modelo TBATS

```
estimator = TBATS(seasonal_periods=(4, 12))
model = estimator.fit(df_tbats_prom_mes)

y_forecast = model.forecast(steps=24)
```

Fuente: Elaboración propia

Gráfico 63: Predicción del modelo TBATS



Fuente: Elaboración propia

A comparación de los modelos vistos anteriormente, TBATS en su predicción muestra ligeros cambios respecto a la tendencia de los datos reales, aun así, no deja de ser un modelo que visualmente presenta una buena predicción.

NNETAR

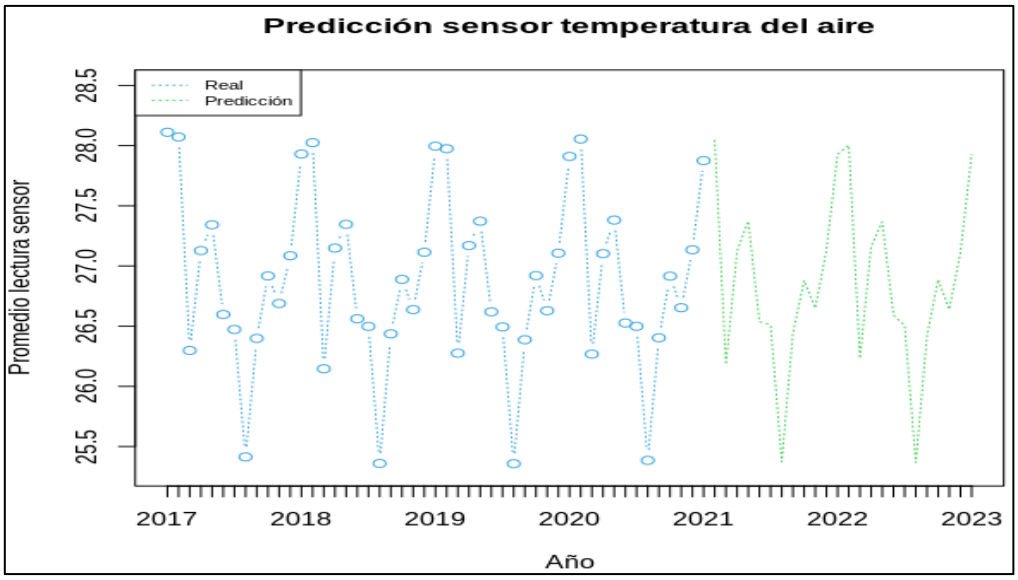
Para la aplicación de este modelo, se ha utilizado el lenguaje de programación R, ya que es exclusivo del mismo y se basa en el uso de redes neuronales para determinar sus predicciones, de igual manera, se trabaja con el mismo conjunto de datos.

Algoritmo 26: Entrenamiento y predicción del modelo NNETAR

```
model_nnetar <- nnetar(ts_m, lambda=0.5)
prediccion_nnetar<- forecast(model_nnetar, h=24)

result <- c(prediccion_nnetar[["x"]], prediccion_nnetar[["mean"]])
ts_nnetar <- ts(result, start = c(2017, 1), frequency = 12)

plot(prediccion_nnetar, col="#3AA5E1" , fcol= "#43CD55", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted",
xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", flwd="b",
xaxt="none", ylim=c(25.3,28.5))
axis(1, at=1:length(index(ts_nnetar)), labels=index(ts_nnetar))
legend("topleft", legend=c("Ventas totales", "Predicción"),
       col=c("#3AA5E1", "#43CD55"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 64: Predicción del modelo NNETAR



Fuente: Elaboración propia

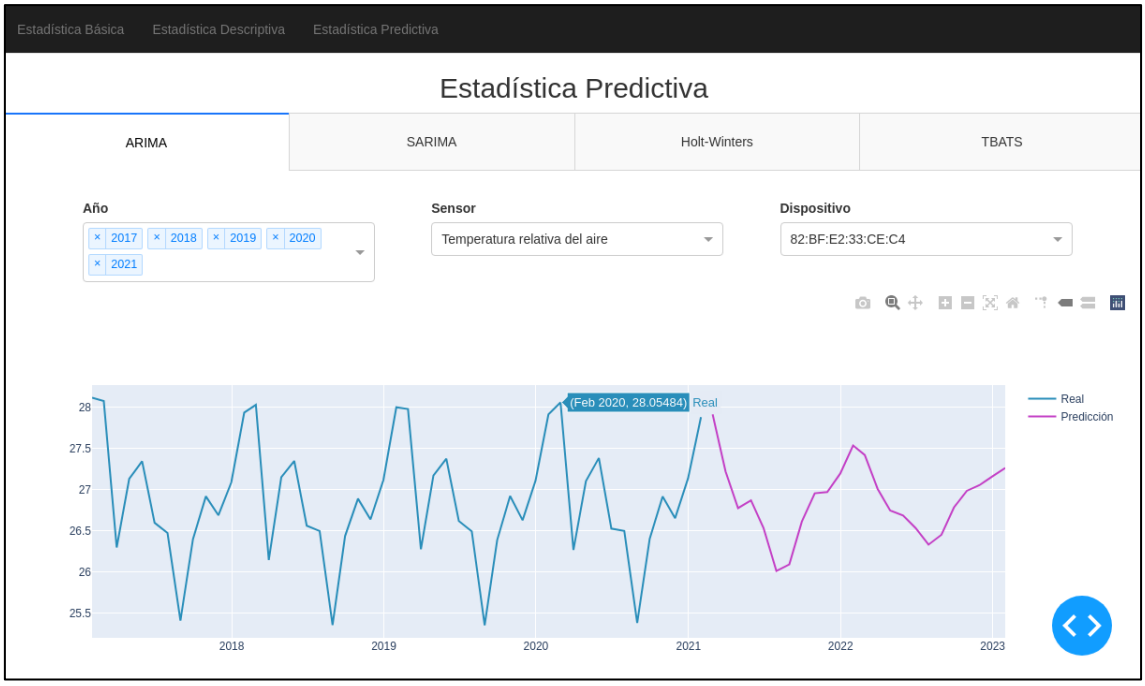
Al igual que los modelos SARIMA, Holt-Winters, STLM y STLF, la predicción del modelo NNETAR, presenta visualmente, una gran similitud respecto a los datos reales. La aplicación de los modelos ARIMA, SARIMA, Holt-Winters y TBATS en R se encuentran en el **Anexo 2**.

Para evaluar qué modelo se adapta mejor al conjunto de datos, el análisis de métricas de calidad para cada modelo se realiza en el Capítulo 3.

2.5.4. Despliegue de resultados sobre un Dashboard

A través de la librería Plotly Dash de Python, se desarrolla un sistema que presente los resultados de aplicar los distintos modelos de análisis seleccionados para este proyecto. Como se aprecia en el **Gráfico 65**, el sistema se categoriza en base a los distintos tipos de análisis que se aplicó, es decir, estadística básica, estadística descriptiva y estadística predictiva.

Gráfico 65: Diseño de Dashboard



Fuente: Elaboración propia

3. CAPÍTULO III: Evaluación del prototipo

La evaluación se divide en dos escenarios, evaluación de integración de datos y evaluación de modelos de predicción en series de tiempo.

3.1. Diseño del experimento

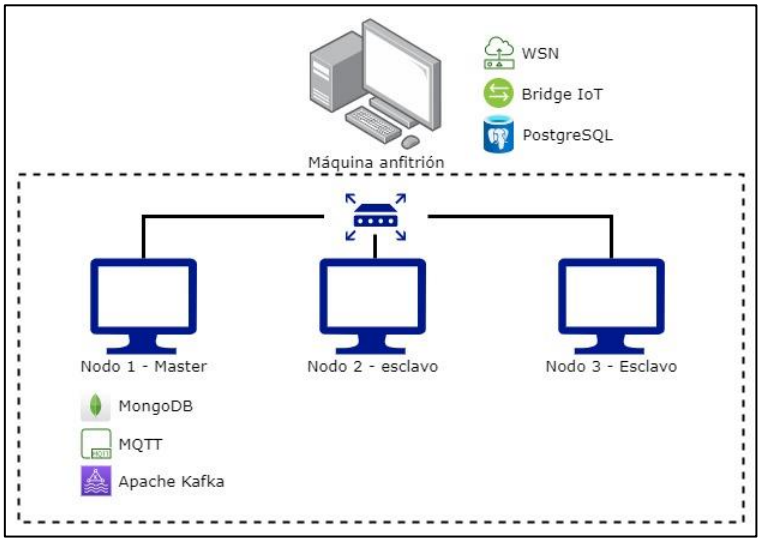
3.1.1. Escenario 1: Evaluación de integración de datos

El escenario 1 está ambientado sobre la capa de redes simuladas hasta la capa de almacenamiento y procesamiento distribuido, como se detalla en el **Gráfico 1** al inicio del Capítulo II. La evaluación de este escenario se subdivide en dos partes:

Captura de datos en streaming

Consiste en evaluar los tiempos de transferencia, en milisegundos, para el proceso de envío de mensajes de lecturas correspondientes a un lote de sensores conectados a un dispositivo, con una frecuencia de envío de 300 ms, que es diferente a la fecha de cada lectura, la cual es de 10 minutos por envió, como se observa en el **Gráfico 66**, todos los sistemas se ejecutan sobre la máquina anfitriona del clúster mientras que los servicios de bases de datos están distribuidos entre el nodo máster y la máquina anfitriona, y los protocolos de IoT están en el nodo master, para ver los recursos de la arquitectura, puede observar la **Tabla 5**.

Gráfico 66: Distribución de servicios - Captura de datos en streaming



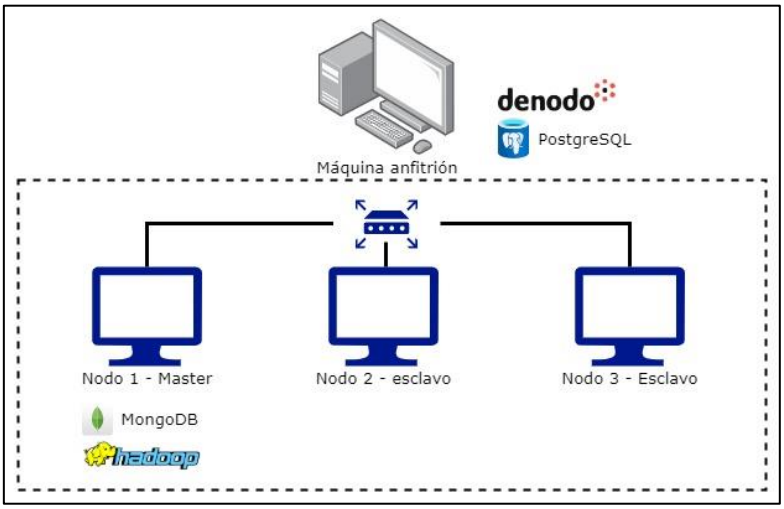
Fuente: Elaboración propia

Para las pruebas se realizaron envíos de 10, 50, 100 y 1000 mensajes de forma individual, los mensajes también son denominados como observaciones a lo largo de esta evaluación.

Integración y virtualización de datos

Comprende la medición de tiempos de carga, en minutos, y el consumo de recursos (CPU y RAM) tanto en la máquina anfitriona como en el nodo master para el proceso de integración y virtualización de datos de PostgreSQL y MongoDB a través de Denodo considerando el conjunto completo de observaciones almacenadas hasta ese momento, el cual fue de 2.5 millones aproximadamente, cada observación compuesta de 22 variables. En el **Gráfico 67** se observa que la máquina anfitriona tiene en ejecución Denodo y PostgreSQL, mientras que el nodo master ejecuta MongoDB y el servicio HDFS del ecosistema Hadoop.

Gráfico 67: Distribución de herramientas – Integración y virtualización de datos



Fuente: Elaboración propia

Las pruebas a realizar están compuestas por diferentes conjuntos de observaciones: 500 mil, 1.5 millones y el conjunto original de 2.5 millones aproximadamente, recalcando que los conjuntos menores de observaciones parten del conjunto original.

3.1.2. Escenario 2: Evaluación de modelos de predicción en series temporales

El escenario 2 consiste en evaluar los modelos de predicción por medio del uso de métricas de calidad, este escenario ejecuta procesos de manera distribuida a

través del servicio YARN del ecosistema Hadoop, se trabaja sobre el conjunto de datos perteneciente a los años 2017 hasta 2021 de lecturas del sensor de temperatura relativa del aire correspondientes al dispositivo 82:BF:E2:33:CE:C4, para ello, todas las herramientas y servicios son ejecutados únicamente en el nodo master utilizando los recursos del clúster.

3.2. Plan de evaluación

3.2.1. Escenario 1: Evaluación de integración de datos

A continuación, se detallan cada una de las variables y su descripción al momento de evaluar las subdivisiones del escenario 1.

Captura de datos en streaming

Tabla 7: Variables - Captura de datos en streaming

Variable	Descripción
T1	Tiempo (ms) que toma el envío de mensajes desde simulador WSN hasta suscriptor MQTT (Bridge IoT)
T2	Tiempo (ms) que toma el envío de mensajes desde productor Apache Kafka (Bridge IoT) hasta consumidor Apache Kafka (Bridge IoT)
T3	Tiempo (ms) que toma el almacenamiento de mensajes que llegan al consumidor Apache Kafka (Bridge IoT) en MongoDB
Total	Suma de los tiempos T1, T2 y T3

Fuente: Elaboración propia

Integración y virtualización de datos

Tabla 8: Variables - Integración y virtualización de datos

Variable	Descripción
T4	Tiempo (min) de ejecución para obtener datos virtualizados desde Denodo
T5	Tiempo (min) de ejecución para almacenar datos virtualizados en HDFS
Total	Suma de los tiempos T4 y T5

Fuente: Elaboración propia

3.2.2. Escenario 2: Evaluación de modelos de predicción en series temporales

Consiste en aplicar métricas que evalúen la calidad de los modelos de series de tiempo a través de las librerías correspondientes en Python y R.

Tabla 9: Métricas de evaluación de calidad de modelos de predicción

Métrica	Fórmula	Librería	
		Python	R
Error medio absoluto	$\text{MAE} = \frac{\sum_{i=0}^{n}\lvert y_i - \hat{y}_i\rvert}{n}$	sklearn.metrics	MLmetrics
Error absoluto mediano	$\text{MedAE}(y, \hat{y}) = \text{median}(\lvert y_1 - \hat{y}_1\rvert, \lvert y_2 - \hat{y}_2\rvert, \ldots, \lvert y_n - \hat{y}_n\rvert)$		
Error cuadrático medio	$\text{MSE} = \frac{\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}{n}$		
Error de porcentaje medio absoluto	$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left\lvert\frac{y_i - \hat{y}_i}{y_i}\right\rvert$		

Fuente: Elaboración propia

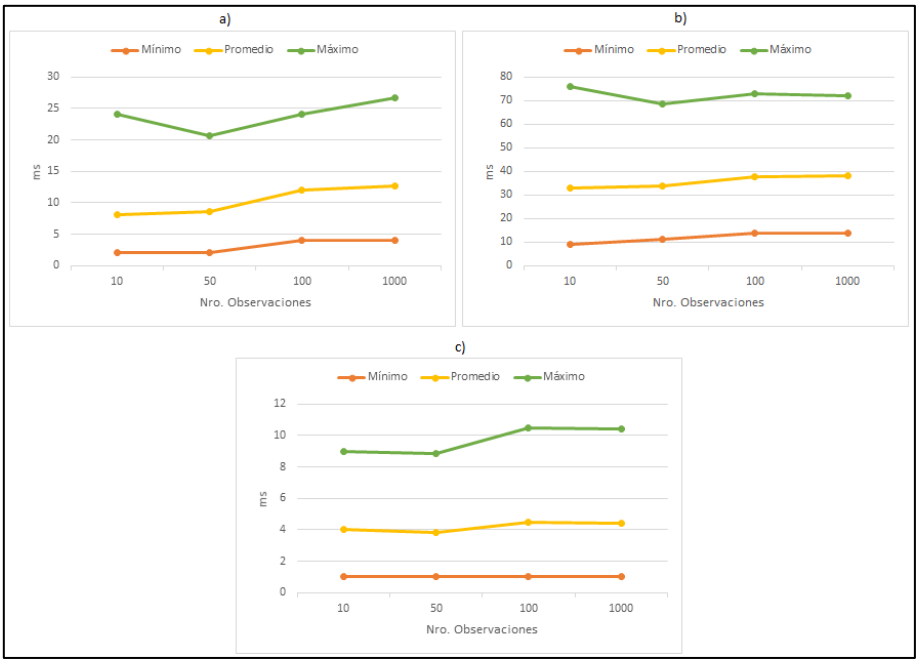
De acuerdo con [80], error medio absoluto, promedio de valores absolutos que permite medir con precisión un modelo de predicción en base a los datos observados y predichos. Error absoluto mediano, robusto con valores atípicos, toma la mediana de las diferencias absolutas. Error cuadrático medio, calcula el promedio de la diferencia cuadrática de los y el error de porcentaje medio absoluto, calcula el tamaño de error absoluto en términos porcentuales.

3.3. Resultados de la evaluación

3.3.1. Escenario 1: Evaluación de integración de datos

Captura de datos en streaming

Gráfico 68: Mínimos, promedios y máximos de - a) T1, b) T2 y c) T3



Fuente: Elaboración propia

Tabla 10: Resultados captura de datos en streaming

Unidad de medida		Milisegundos			
Observaciones		**T1**	**T2**	**T3**	**Total**
10	**Mínimo**	2	9	1	12
	Promedio	6.1	24	3	33.1
	Máximo	16	43	5	64
50	**Mínimo**	2	11	1	14
	Promedio	6.62	22.76	2.82	32.2
	Máximo	12	35	5	52
100	**Mínimo**	4	14	1	19
	Promedio	8	23.76	3.45	35.21
	Máximo	12	35	6	53
1000	**Mínimo**	4	14	1	19
	Promedio	8.64	24.06	3.42	36.12
	Máximo	14	34	6	54

Fuente: Elaboración propia

De acuerdo con la **Tabla 11**, a medida que se incrementó el número de observaciones, se incrementaron los promedios de tiempo de envío para todas las variables, la variable que más destaca por representar una mayor cantidad es T2 con tiempos de envío entre 20 y 25 milisegundos, mientras que la variable que representa el menor tiempo de envío es T3 con valores entre 2.5 y 3.5 milisegundos.

Integración y virtualización de datos

Tabla 11: Resultados integración y virtualización de datos

Unidad de medida		Minutos		
Observaciones		**T4**	**T5**	**Total**
500 000	**Mínimo**	02:33.179	00:10.402	02:43.581
	Promedio	02:40.205	00:11.368	02:51.573
	Máximo	02:51.524	00:12.773	03:04.297
1 500 000	**Mínimo**	05:26.476	00:30.925	05:57.401
	Promedio	05:31.343	00:35.741	06:07.084
	Máximo	05:35.383	00:39.921	06:15.304
2 524 614	**Mínimo**	08:56.091	00:56.734	09:52.825
	Promedio	08:59.372	00:59.763	09:59.135
	Máximo	09:03.266	01:05.549	10:08.815

Fuente: Elaboración propia

Los resultados indican notablemente que, a una mayor cantidad de observaciones acumuladas, mucho mayor va a ser el tiempo que tarde en integrar y virtualizar los datos Denodo (T4), mientras que es considerablemente menor el tiempo de retraso para almacenar los datos virtualizados en HDFS (T5).

Tabla 12: Consumo de recursos de integración y virtualización de datos

Máquina		**Anfitriona**	**Master**	**Anfitriona**	**Master**
Observaciones		**CPU (%)**		**RAM (GB)**	
500 000	**Mínimo**	-	-	1.30	1.36
	Promedio	-	-	1.57	1.45
	Máximo	-	-	1.80	1.53
	Valor neto	29 – 44	33 - 41		
1 500 000	**Mínimo**	-	-	1.30	3.35
	Promedio	-	-	1.57	3.46
	Máximo	-	-	1.80	3.54
	Valor neto	30 – 49	31 – 45		
2 524 614	**Mínimo**	-	-	1.50	5.96
	Promedio	-	-	1.64	6.04
	Máximo	-	-	1.80	6.17
	Valor neto	35 – 57	27 – 62		

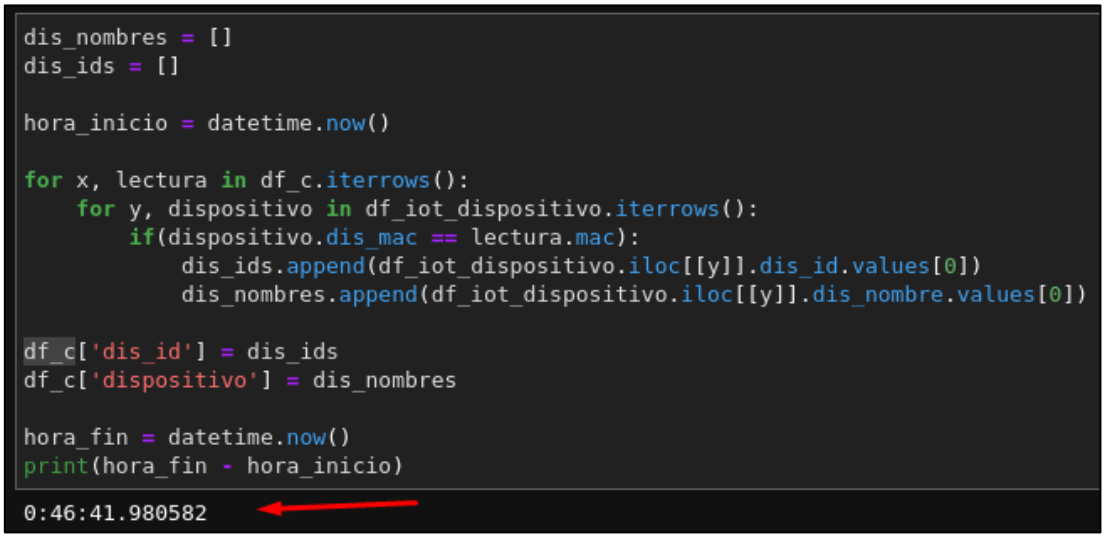
Fuente: Elaboración propia

Adicionalmente, se analiza el consumo de recursos tanto en la máquina anfitriona como en el nodo master, respecto al consumo de CPU, se pudo notar un gran incremento de un 30% aproximadamente como mínimo para todo el conjunto de observaciones, mientras que el valor máximo dependía de su tamaño, es decir, a mayor cantidad de observaciones acumuladas, mayor consumo de CPU y en escasas ocasiones para todo el conjunto de observaciones, el valor máximo rondaba el 70% y 80% aproximadamente, todos los valores aplican tanto en la máquina anfitriona como el nodo master. Hay que considerar que esta medición no es precisa, por lo que el consumo de CPU fluctúa demasiado dependiendo de los procesos adicionales que se estén ejecutando. Por otro lado, el consumo de RAM fue distinto, para la máquina anfitriona, el consumo era constante, no variaba mucho para todos los conjuntos de observaciones, el incremento de RAM era de 1.6 GB aproximadamente, mientras que para el nodo master, si se nota un incremento en función del tamaño del conjunto de observaciones, llegando a consumir un promedio de 6

GB para el mayor conjunto, mientras que, para el menor conjunto, fue un incremento promedio de 1.45 GB.

Respecto al proceso de integración de datos, al comparar el mismo proceso a través del uso de lógica de programación y al ser un proceso simple que se basa en extraer los conjuntos de datos tanto de MongoDB como PostgreSQL y relacionarlos por medio de algoritmos de iteración (Ver **Gráfico 69**), tan solo con aplicar la integración con una tabla de PostgreSQL y sin realizar el proceso de aplanar los datos en MongoDB, el tiempo de ejecución fue de 45 minutos aproximadamente para un conjunto de datos de 600 mil y 8 datos de lecturas y dispositivos respectivamente, algo que a comparación con la herramienta Denodo es muy diferente, ya que los resultados de evaluación del escenario 1 para la integración de datos promediaron los 10 minutos aproximadamente, esto se debe a la falta de conocimiento sobre los lenguajes y cómo aplicar correctamente algoritmos para lograr una integración eficiente, además la herramienta Denodo no tiene la necesidad de aplicar comandos, tal como se evidencia durante el ensamblaje del prototipo, la integración se la realiza de manera gráfica y en mucho menor tiempo, lo contrario que requeriría una investigación profunda sobre algoritmos y lógica de programación.

Gráfico 69: Tiempo retraso de integración utilizando lógica de programación



```
dis_nombres = []
dis_ids = []

hora_inicio = datetime.now()

for x, lectura in df_c.iterrows():
    for y, dispositivo in df_iot_dispositivo.iterrows():
        if(dispositivo.dis_mac == lectura.mac):
            dis_ids.append(df_iot_dispositivo.iloc[[y]].dis_id.values[0])
            dis_nombres.append(df_iot_dispositivo.iloc[[y]].dis_nombre.values[0])

df_c['dis_id'] = dis_ids
df_c['dispositivo'] = dis_nombres

hora_fin = datetime.now()
print(hora_fin - hora_inicio)
```

```
0:46:41.980582
```

Fuente: Elaboración propia

La evidencia de las pruebas realizadas para este escenario se encuentra en el **Anexo 3**.

3.3.2. Escenario 2: Evaluación de modelos de predicción en series temporales

Tabla 13: Resultados de aplicación de métricas de evaluación de calidad

Modelo	Lenguaje	Métrica de calidad			
		MAE	MedAE	MSE	MAPE
ARIMA	Python	0,3924	0,3786	0,2423	0,0147
	R	0,3928	0,3791	0,2425	0,0147
SARIMA	Python	0,0659	0,0578	0,0061	0,0024
	R	0,0259	0,0178	0,0011	0,0010
Holt-Winters	Python	0,0472	0,0512	0,0030	0,0018
	R	0,0271	0,0252	0,0011	0,0010
TBATS	Python	0,2335	0,1893	0,0833	0,0087
	R	0,0250	0,0247	0,0010	0,0009
STLM	R	0,0254	0,0218	0,0009	0,0009
STLF	R	0,0274	0,0231	0,0010	0,0010
NNETAR	R	0,0395	0,0305	0,0023	0,0015
Mejor modelo		TBATS - R	SARIMA - R	STLM	STLM

Fuente: Elaboración propia

De acuerdo con la información de los resultados en la **Tabla 12**, al aplicar las métricas de evaluación de calidad sobre los conjuntos de datos reales y datos predichos por los modelos, se determina que los modelos que mejor se adaptan al conjunto de datos y realizan una predicción más precisa son STLM y TBATS en R, ya que en comparación con los resultados de los otros modelos, son los que menor margen de error tuvieron en dos de las métricas aplicadas y su cercanía a los mejores resultados en las métricas restantes, aún así, realizando una comparación gráfica de los modelos, a excepción de ARIMA y TBATS en Python, todos presentan una gran similitud en su predicción, las diferencias al aplicar las métricas de calidad fueron mínimas.

CONCLUSIONES

La implementación de tecnologías de IoT ha innovado el campo de la agricultura, a través del seguimiento y monitoreo de cultivos, cada uno de los procesos detallados en este proyecto ayudan a cumplir el objetivo de mejorar la calidad de producción, y para el Ecuador sería una alternativa que refuerce mucho las labores del agricultor y ayude al crecimiento económico del país.

El proceso de integración de datos de IoT en el contexto de agricultura de precisión implica realizar una serie de actividades, desde la recolección de datos, específicamente para este proyecto, a través del uso de un simulador WSN, desarrollo de un middleware IoT que gestione la transferencia y almacenamiento de mensajes generados por la WSN y finalmente el uso de Denodo como herramienta de integración y virtualización de datos desde distintas fuentes de información. La selección de Denodo como herramienta de integración de datos surge a comparación de un proceso manual de integración con conocimientos intermedios de programación, y como resultados, Denodo destaca por su facilidad de uso, velocidad de procesamiento, múltiples conexiones a fuentes de información y el poco tiempo que requirió realizar todo el proceso.

Big Data significa grandes volúmenes de datos, y este surge al integrar información agrícola con diversos puntos de lecturas de datos, eso provoca la necesidad de herramientas de almacenamiento y procesamiento distribuido como lo es Apache Hadoop, cuya importancia como plataforma de Big Data para este proyecto ha sido reflejada en los procesos de integración y análisis de datos.

Los modelos de predicción en series temporales se evaluaron utilizando métricas de calidad con la finalidad de obtener el modelo que mejor se adapte al conjunto de datos observados, los resultados arrojados fueron muy ajustados y el modelo que menor margen de error tuvo fueron STLM y TBATS en R. Por otro lado, la evaluación de tiempos de transferencia de mensajes en streaming fueron positivos por el bajo promedio de tiempo, sin extenderse más allá de los 40 milisegundos.

RECOMENDACIONES

Contemplar de mejor manera el comportamiento de factores ambientales en un cultivo, permitirá que el simulador WSN genere datos mucho más reales y pueda ser utilizado como una herramienta alternativa para el proceso de adquisición de datos, pero es mucho más recomendable realmente trabajar con datos generados en escenario real.

El problema de herramientas como Denodo es que no son gratuitas, existe cierta restricción para algunas funcionalidades, por ejemplo, la conexión a MongoDB requirió la adquisición de un driver especial, y para ello, se recomienda la búsqueda alternativa de herramientas o crear un propio proceso de integración que obtenga resultados similares o mejores a herramientas ya existentes.

Una mayor asignación de recursos a los equipos permite un mejor rendimiento del clúster Hadoop para el almacenamiento y procesamiento distribuido de grandes cantidades de datos, y mucho mejor si es una arquitectura destinada solo a ese uso, es decir, distribuir los distintos servicios y herramientas adicionales, en máquinas que no pertenezcan a la arquitectura del clúster, para el desarrollo de este prototipo, una de las limitantes fue esa, toda la arquitectura fue puesta en ejecución en una sola máquina con bajos recursos considerando que es un proyecto de ciencia de datos y la cantidad de datos que se analizaron fueron de aproximadamente 2.5 millones.

Es posible aplicar otros métodos de análisis para grandes volúmenes de información, dentro de la minería de datos existen otras ramificaciones más avanzadas que implican Machine learning o Deep learning, las cuales, si se profundizan, podrían brindar resultados más precisos de cara a obtener información significativa para el proceso de toma de decisiones en el campo de la agricultura de precisión.

BIBLIOGRAFÍA

[1] Nandrade, «La Importancia de la Agricultura en nuestro país». https://www.utn.edu.ec/ficaya/carreras/agropecuaria/?p=1091 (accedido abr. 16, 2021).

[2] «FAO - Noticias: 2050: un tercio más de bocas que alimentar». http://www.fao.org/news/story/es/item/35675/icode/ (accedido feb. 17, 2021).

[3] G.-A. Musat *et al.*, «Advanced services for efficient management of smart farms», *J. Parallel Distrib. Comput.*, vol. 116, pp. 3-17, jun. 2018, doi: 10.1016/j.jpdc.2017.10.017.

[4] M. Colezea, G. Musat, F. Pop, C. Negru, A. Dumitrascu, y M. Mocanu, «CLUeFARM: Integrated web-service platform for smart farms», *Comput. Electron. Agric.*, vol. 154, pp. 134-154, nov. 2018, doi: 10.1016/j.compag.2018.08.015.

[5] J. Berrú-Ayala, D. Hernandez, P. Morocho-Díaz, J. P. N. Vicuña, B. Mazon-Olivo, y A. Pan, «SCADA System Based on IoT for Intelligent Control of Banana Crop Irrigation», 2020, pp. 243-256.

[6] U. S. Rajani, A. Sathyan, A. Mohan, y A. A. Kadar, «Design architecture of autonomous precision farming system», en *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*, jul. 2017, pp. 415-419, doi: 10.1109/ICICICT1.2017.8342599.

[7] I. Ramírez Morales, B. Mazon Olivo, y A. Pan, «Ciencia de datos en el sector agropecuario», Machala : Universidad Técnica de Machala, 2018.

[8] P. Balogh *et al.*, «Main Motivational Factors of Farmers Adopting Precision Farming in Hungary», *Agronomy*, vol. 10, n.º 4, Art. n.º 4, abr. 2020, doi: 10.3390/agronomy10040610.

[9] Y. Vecchio, G. P. Agnusdei, P. P. Miglietta, y F. Capitanio, «Adoption of Precision Farming Tools: The Case of Italian Farmers», *Int. J. Environ. Res. Public. Health*, vol. 17, n.º 3, Art. n.º 3, ene. 2020, doi: 10.3390/ijerph17030869.

[10] «Agricultura de Precisión». http://www.grap.udl.cat/es/presentacion/ap.html (accedido feb. 25, 2021).

[11] S.-J. Hwang, J.-Y. Lee, y J.-S. Nam, «Irrigation System for a Roller-Type Onion Pot Seeding Machine», *Appl. Sci.*, vol. 9, n.º 3, Art. n.º 3, ene. 2019, doi: 10.3390/app9030430.

[12] M. F. Işık, Y. Sönmez, C. Yılmaz, V. Özdemir, y E. N. Yılmaz, «Precision Irrigation System (PIS) Using Sensor Network Technology Integrated with IOS/Android Application», *Appl. Sci.*, vol. 7, n.º 9, Art. n.º 9, sep. 2017, doi: 10.3390/app7090891.

[13] R. S. Krishnan *et al.*, «Fuzzy Logic based Smart Irrigation System using Internet of Things», *J. Clean. Prod.*, vol. 252, p. 119902, abr. 2020, doi: 10.1016/j.jclepro.2019.119902.

[14] D. Hernández Rojas, B. Mazon Olivo, y C. Escudero, «Internet de las cosas (IoT)», Machala : Universidad Técnica de Machala, 2018.

[15] M. Saqib, T. A. Almohamad, y R. M. Mehmood, «A Low-Cost Information Monitoring System for Smart Farming Applications», *Sensors*, vol. 20, n.º 8, Art. n.º 8, ene. 2020, doi: 10.3390/s20082367.

[16] J. Novillo-Vicuña, D. Hernández Rojas, B. Mazón Olivo, J. Molina Ríos, y O. Cárdenas Villavicencio, *Arduino y el Internet de las cosas*, 1.ª ed. Editorial Científica 3Ciencias, 2018.

[17] D. Vergnaud, «Comment on "Efficient and Secure Outsourcing Scheme for RSA Decryption in Internet of Things"», *IEEE Internet Things J.*, vol. 7, n.º 11, pp. 11327-11329, nov. 2020, doi: 10.1109/JIOT.2020.3004346.

[18] J. Kua, S. H. Nguyen, G. Armitage, y P. Branch, «Using Active Queue Management to Assist IoT Application Flows in Home Broadband Networks», *IEEE Internet Things J.*, vol. 4, n.º 5, pp. 1399-1407, oct. 2017, doi: 10.1109/JIOT.2017.2722683.

[19] A. Moon, J. Kim, J. Zhang, y S. W. Son, «Evaluating fidelity of lossy compression on spatiotemporal data from an IoT enabled smart farm», *Comput. Electron. Agric.*, vol. 154, pp. 304-313, nov. 2018, doi: 10.1016/j.compag.2018.08.045.

[20] A. Bhatt, «A Quick Guide to Understanding IoT Application Messaging Protocols», jun. 27, 2018. https://www.einfochips.com/blog/a-quick-guide-to-understanding-iot-application-messaging-protocols/ (accedido mar. 02, 2021).

[21] X. Liu, T. Zhang, N. Hu, P. Zhang, y Y. Zhang, «The method of Internet of Things access and network communication based on MQTT», *Comput. Commun.*, vol. 153, pp. 169-176, mar. 2020, doi: 10.1016/j.comcom.2020.01.044.

[22] B. Mishra y A. Kertesz, «The Use of MQTT in M2M and IoT Systems: A Survey», *IEEE Access*, vol. 8, pp. 201071-201086, 2020, doi: 10.1109/ACCESS.2020.3035849.

[23] «MQTT - The Standard for IoT Messaging». https://mqtt.org/ (accedido mar. 02, 2021).

[24] B. Mazon-Olivo, D. Hernández-Rojas, J. Maza-Salinas, y A. Pan, «Rules engine and complex event processor in the context of internet of things for precision agriculture», *Comput. Electron. Agric.*, vol. 154, pp. 347-360, nov. 2018, doi: 10.1016/j.compag.2018.09.013.

[25] H. Wu, «Research Proposal: Reliability Evaluation of the Apache Kafka Streaming System», en *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, oct. 2019, pp. 112-113, doi: 10.1109/ISSREW.2019.00055.

[26] J. Bang, S. Son, H. Kim, Y. Moon, y M. Choi, «Design and implementation of a load shedding engine for solving starvation problems in Apache Kafka», en *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, abr. 2018, pp. 1-4, doi: 10.1109/NOMS.2018.8406306.

[27] A. Almasoud, H. Al-Khalifa, A. Al-salman, y M. Lytras, «A Framework for Enhancing Big Data Integration in Biological Domain Using Distributed Processing», *Appl. Sci.*, vol. 10, n.º 20, Art. n.º 20, ene. 2020, doi: 10.3390/app10207092.

[28] «Denodo Platform Overview», *Denodo*, mar. 03, 2017. https://www.denodo.com/en/denodo-platform/overview (accedido mar. 08, 2021).

[29] C. A. Győrödi, D. V. Dumşe-Burescu, D. R. Zmaranda, R. Ş. Győrödi, G. A. Gabor, y G. D. Pecherle, «Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage», *Appl. Sci.*, vol. 10, n.º 23, Art. n.º 23, ene. 2020, doi: 10.3390/app10238524.

[30] C. Rong, W. Bin, y Y. Chuanxu, «Design of Heterogeneous Data SQL Access Scheme Based on PostgreSQL», en *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, oct. 2019, pp. 1520-1524, doi: 10.1109/EITCE47263.2019.9094863.

[31] «PostgreSQL: About». https://www.postgresql.org/about/ (accedido mar. 08, 2021).

[32] C. Moreno *et al.*, «RiverCore: IoT Device for River Water Level Monitoring over Cellular Communications», *Sensors*, vol. 19, n.º 1, Art. n.º 1, ene. 2019, doi: 10.3390/s19010127.

[33] A. M. Campoverde, D. L. H. R, y B. E. M. O, «Cloud computing con herramientas open-source para Internet de las cosas», *Maskana*, vol. 6, pp. 173-182, dic. 2015.

[34] B. Huang *et al.*, «RDMA-driven MongoDB: An approach of RDMA enhanced NoSQL paradigm for large-Scale data processing», *Inf. Sci.*, vol. 502, pp. 376-393, oct. 2019, doi: 10.1016/j.ins.2019.06.048.

[35] J. E. Newton, R. Nettle, y J. E. Pryce, «Farming smarter with big data: Insights from the case of Australia's national dairy herd milk recording scheme», *Agric. Syst.*, vol. 181, p. 102811, may 2020, doi: 10.1016/j.agsy.2020.102811.

[36] T. Wilcox, N. Jin, P. Flach, y J. Thumim, «A Big Data platform for smart meter data analytics», *Comput. Ind.*, vol. 105, pp. 250-259, feb. 2019, doi: 10.1016/j.compind.2018.12.010.

[37] B. Mazon-Olivo, A. Pan, y R. Tinoco-Egas, «Inteligencia de negocios en el sector agropecuario», Machala : Universidad Técnica de Machala, 2018.

[38] A. M. Ali, *Big Data Analytics in HIV/AIDS Research*. IGI Global, 2018.
[39] H.-C. Lu, F. J. Hwang, y Y.-H. Huang, «Parallel and distributed architecture of genetic algorithm on Apache Hadoop and Spark», *Appl. Soft Comput.*, vol. 95, p. 106497, oct. 2020, doi: 10.1016/j.asoc.2020.106497.
[40] T. H. Aung y W. T. Zaw, «Improved Job Scheduling for Achieving Fairness on Apache Hadoop YARN», en *2020 International Conference on Advanced Information Technologies (ICAIT)*, nov. 2020, pp. 188-193, doi: 10.1109/ICAIT51105.2020.9261793.
[41] J. Zhang, Z. Ye, y K. Zheng, «A Parallel Computing Approach to Spatial Neighboring Analysis of Large Amounts of Terrain Data Using Spark», *Sensors*, vol. 21, n.º 2, Art. n.º 2, ene. 2021, doi: 10.3390/s21020365.
[42] «What is HDFS? Apache Hadoop Distributed File System», ago. 25, 2020. https://www.ibm.com/analytics/hadoop/hdfs (accedido mar. 09, 2021).
[43] «Apache Hadoop 2.7.2 – HDFS Architecture». https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html (accedido mar. 09, 2021).
[44] T. Yeh y H. Huang, «Realizing integrated prioritized service in the Hadoop cloud system», *Future Gener. Comput. Syst.*, vol. 100, pp. 176-185, nov. 2019, doi: 10.1016/j.future.2019.05.012.
[45] «Apache Hadoop 3.3.0 – Apache Hadoop YARN». https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html (accedido mar. 09, 2021).
[46] V. S. Martha, W. Zhao, y X. Xu, «h-MapReduce: A Framework for Workload Balancing in MapReduce», en *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, mar. 2013, pp. 637-644, doi: 10.1109/AINA.2013.48.
[47] Y. Xu, H. Liu, y Z. Long, «A distributed computing framework for wind speed big data forecasting on Apache Spark», *Sustain. Energy Technol. Assess.*, vol. 37, p. 100582, feb. 2020, doi: 10.1016/j.seta.2019.100582.
[48] «Cluster Mode Overview - Spark 3.1.1 Documentation». https://spark.apache.org/docs/latest/cluster-overview.html (accedido mar. 09, 2021).
[49] «Apache Flink: What is Apache Flink? — Architecture». https://flink.apache.org/flink-architecture.html (accedido mar. 09, 2021).
[50] M. Syafrudin, N. L. Fitriyani, D. Li, G. Alfian, J. Rhee, y Y.-S. Kang, «An Open Source-Based Real-Time Data Processing Architecture Framework for Manufacturing Sustainability», *Sustainability*, vol. 9, n.º 11, Art. n.º 11, nov. 2017, doi: 10.3390/su9112139.
[51] e-Z. Solutions, «Cloud-based Big Data Analytics». https://www.e-zest.com/cloud-based-big-data-analytics (accedido mar. 09, 2021).
[52] O. Haffner, E. Kučera, y M. Moravčík, «Sales Prediction of Svijany Slovakia, Ltd. Using Microsoft Azure Machine Learning and ARIMA», en *2020 Cybernetics Informatics (K I)*, ene. 2020, pp. 1-9, doi: 10.1109/KI48306.2020.9039875.
[53] «¿Qué es AWS?», *Amazon Web Services, Inc.* https://aws.amazon.com/es/what-is-aws/ (accedido mar. 09, 2021).
[54] «What is Google Cloud Platform (GCP)? - Definition from WhatIs.com», *SearchCloudComputing*. https://searchcloudcomputing.techtarget.com/definition/Google-Cloud-Platform (accedido mar. 09, 2021).
[55] «¿Qué es la plataforma IBM Cloud?» https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform (accedido mar. 09, 2021).
[56] B. Mazon-Olivo, M. Pinta, y F. Redrovan, «Desarrollo de competencias en Minería de Datos, una experiencia didáctica», 2020, pp. 383-406.

[57] A. Londhe y P. P. Rao, «Platforms for big data analytics: Trend towards hybrid era», en *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, ago. 2017, pp. 3235-3238, doi: 10.1109/ICECDS.2017.8390056.

[58] «¿Qué es la media, la mediana y la moda?», *QuestionPro*, dic. 09, 2018. https://www.questionpro.com/blog/es/la-media-la-mediana-y-la-moda/ (accedido abr. 04, 2021).

[59] «Varianza y desviación estándar, ejemplos y ejercicios | Matemóvil», oct. 01, 2019. https://matemovil.com/varianza-y-desviacion-estandar-ejemplos-y-ejercicios/ (accedido abr. 04, 2021).

[60] «Coeficiente de variación», *Economipedia*, oct. 02, 2017. https://economipedia.com/definiciones/coeficiente-de-variacion.html (accedido abr. 04, 2021).

[61] «Symmetry, Skewness and Kurtosis | Real Statistics Using Excel». https://www.real-statistics.com/descriptive-statistics/symmetry-skewness-kurtosis/ (accedido abr. 04, 2021).

[62] W. Koehrsen, «Histograms and Density Plots in Python», *Medium*, mar. 23, 2018. https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0 (accedido abr. 04, 2021).

[63] «Box plot review (article)», *Khan Academy*. https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/box-plot-review (accedido abr. 04, 2021).

[64] C. Maklin, «ARIMA Model Python Example — Time Series Forecasting», *Medium*, jul. 21, 2019. https://towardsdatascience.com/machine-learning-part-19-time-series-and-autoregressive-integrated-moving-average-model-arima-c1005347b0d7 (accedido abr. 04, 2021).

[65] J. Brownlee, «A Gentle Introduction to SARIMA for Time Series Forecasting in Python», *Machine Learning Mastery*, ago. 16, 2018. https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/ (accedido abr. 04, 2021).

[66] E. Khan, «Holt-Winters Forecasting», *Medium*, ene. 05, 2021. https://medium.com/analytics-vidhya/holt-winters-forecasting-13c2e60d983f (accedido abr. 04, 2021).

[67] *6.6 STL decomposition | Forecasting: Principles and Practice (2nd ed)*. .

[68] G. Skorupa, «Forecasting Time Series with Multiple Seasonalities using TBATS in Python», *Medium*, ene. 14, 2019. https://medium.com/intive-developers/forecasting-time-series-with-multiple-seasonalities-using-tbats-in-python-398a00ac0e8a (accedido abr. 04, 2021).

[69] «Neural Network Models for Time Series Prediction». https://rstudio-pubs-static.s3.amazonaws.com/313595_2cd55f37d9214b15ab60d7b1a0663da8.html (accedido abr. 04, 2021).

[70] P. Skoda y F. Adam, *Knowledge Discovery in Big Data from Astronomy and Earth Observation: Astrogeoinformatics*. Elsevier, 2020.

[71] B. Mazon-Olivo, O. Romero-Hidalgo, A. Borja-Herrera, M. Aguirre-Benalcazar, M. Contento-Segarra, y M. Jaramillo, «Tecnologías de Inteligencia de Negocios y Minería de datos para el análisis de la producción y comercialización de cacao. Business Intelligence and Data Mining Technologies for the analysis of cocoa production and commercialization», *Espacios*, vol. 39, p. 6, ago. 2018.

[72] «Metodología de base para Análisis Online». https://elartedemedir.com/blog/metodologia-analisis-online/ (accedido abr. 05, 2021).

[73] J. A. Rosiene y C. P. Rosiene, «SPAM: Simplifying Python for Approaching Machine Learning», en *2020 IEEE Frontiers in Education Conference (FIE)*, oct. 2020, pp. 1-3, doi: 10.1109/FIE44824.2020.9273972.

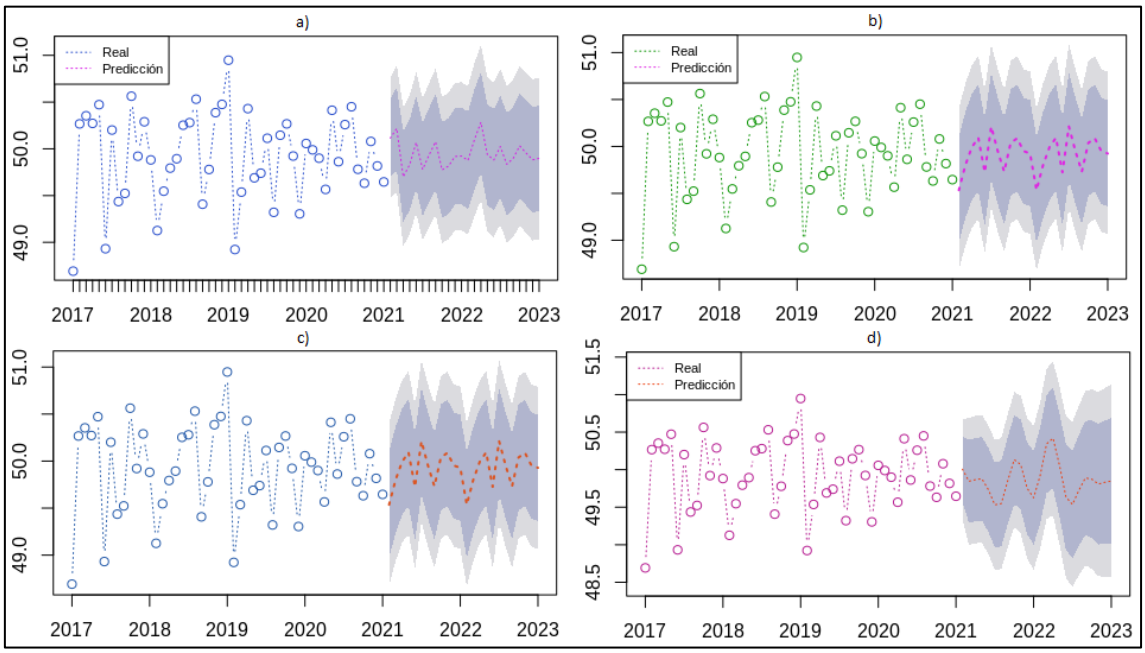
[74] «¿Qué es JavaScript? - Aprende sobre desarrollo web | MDN». https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accedido mar. 11, 2021).
[75] D. Laksono, «Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS Fullstack Web App», en *2018 4th International Conference on Science and Technology (ICST)*, ago. 2018, pp. 1-5, doi: 10.1109/ICSTC.2018.8528705.
[76] «Anaconda Navigator — Anaconda documentation». https://docs.anaconda.com/anaconda/navigator/ (accedido mar. 11, 2021).
[77] J. M. Perkel, «Why Jupyter is data scientists' computational notebook of choice», *Nature*, vol. 563, n.º 7729, Art. n.º 7729, oct. 2018, doi: 10.1038/d41586-018-07196-1.
[78] V. Paz y K. Adrián, «Implementación de herramientas de gestión de LOGS, integración de datos y seguridad para el sistema IOTMACH», nov. 2016, Accedido: abr. 12, 2021. [En línea]. Disponible en: http://repositorio.utmachala.edu.ec/handle/48000/7660.
[79] W. Belduma, «Implementación de una plataforma de big data para la Gestión Hospitalaria», 2020, Accedido: mar. 09, 2021. [En línea]. Disponible en: http://repositorio.utmachala.edu.ec/handle/48000/15976.
[80] joydeep bhattacharjee, «Common metrics for Time Series Analysis», *Medium*, dic. 29, 2019. https://joydeep31415.medium.com/common-metrics-for-time-series-analysis-f3ca4b29fe42 (accedido abr. 12, 2021).

ANEXOS

Anexo 1: Aplicación de modelos con distintos conjuntos de datos

- **Dispositivo: EE:DD:11:55:00:02**
  - **Sensor: Humedad del suelo**

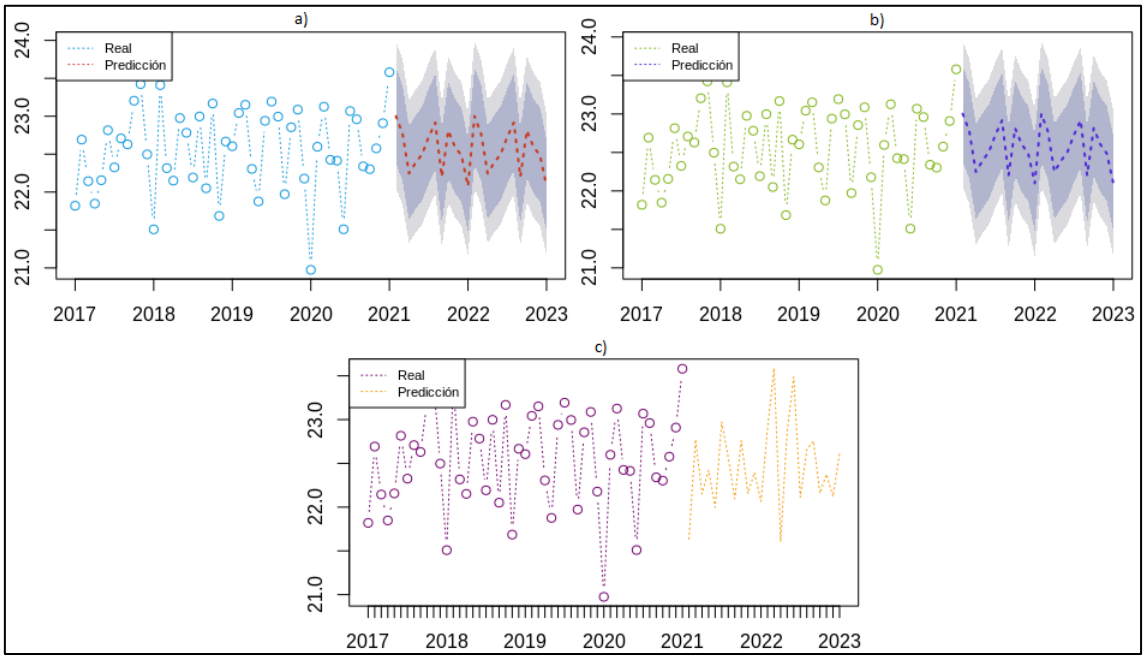
Gráfico 70: a) ARIMA, b) SARIMA, c) Holt-Winters y d) TBATS



Fuente: Elaboración propia

  - **Sensor: Temperatura del suelo**

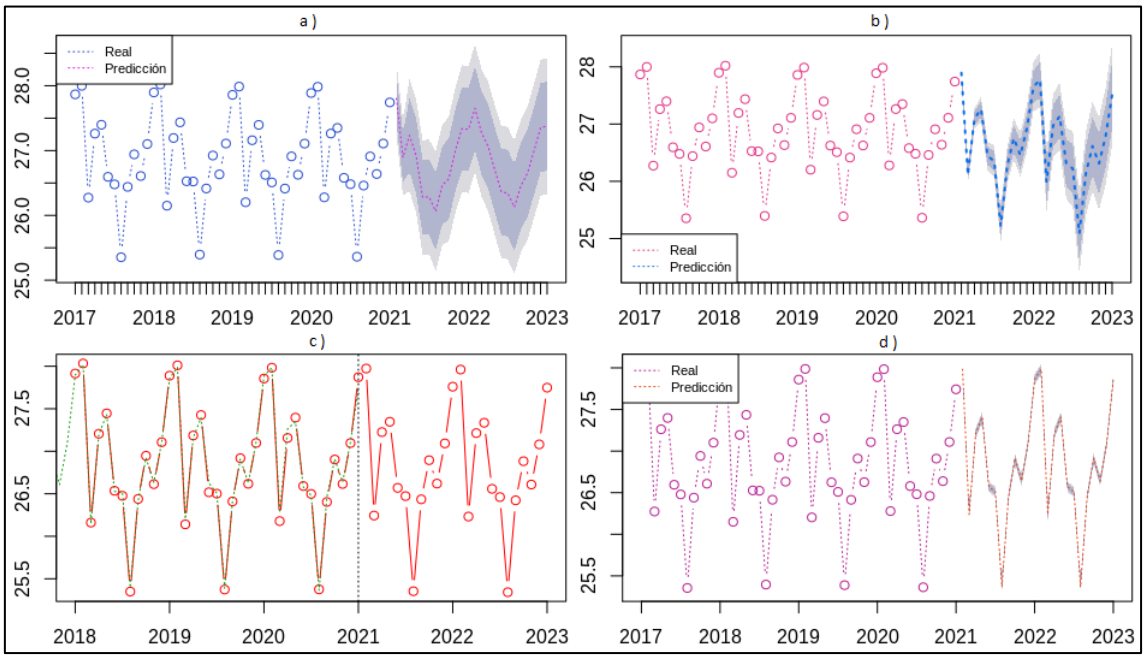
Gráfico 71: a) LSTM, b) LSTF y c) NNETAR



Fuente: Elaboración propia

- **Dispositivo: 92:C8:92:9B:32:D5**
  - **Sensor: Temperatura relativa del aire**

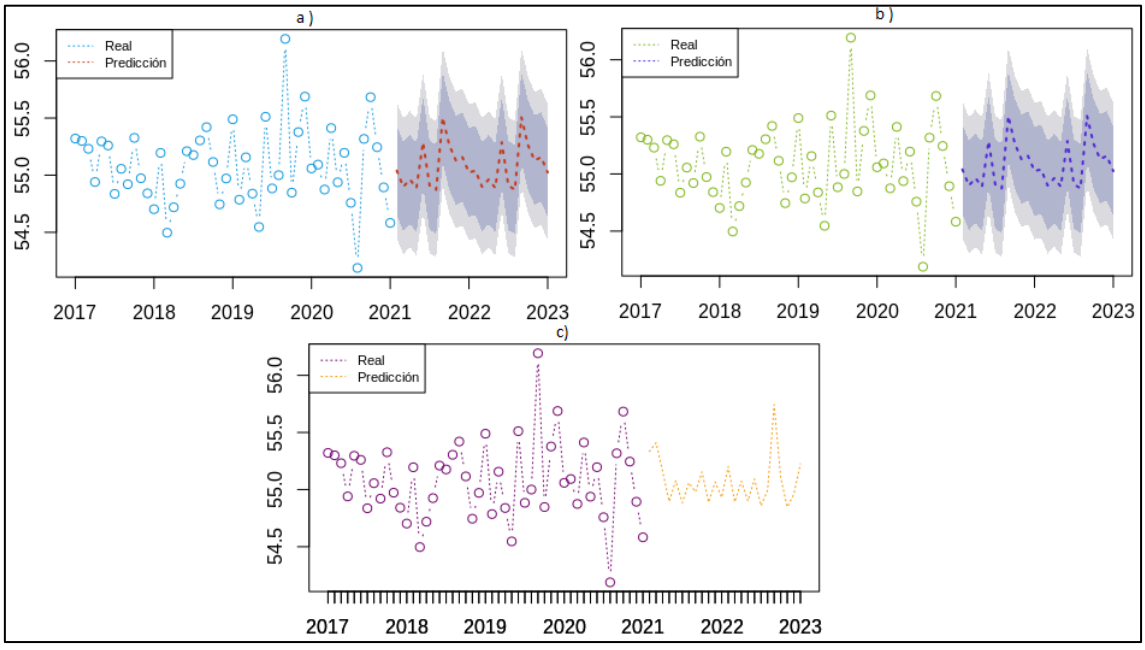
Gráfico 72: a) ARIMA, b) SARIMA, c) Holt-Winters y d) TBATS



Fuente: Elaboración propia

  - **Sensor: Humedad relativa del aire**

Gráfico 73: a) LSTM, b) LSTF y c) NNETAR



Fuente: Elaboración propia

Anexo 2: Modelos ARIMA, SARIMA, Holt-Winters y TBATS en R

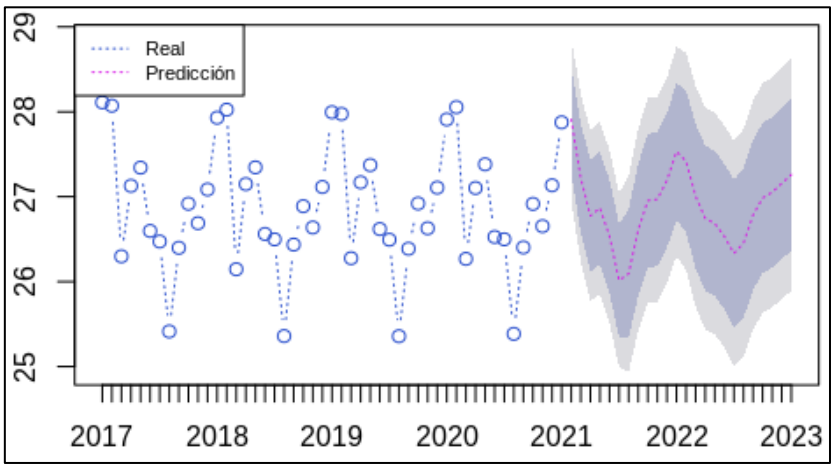
- **Dispositivo: 82:BF:E2:33:CE:C4, Sensor: Temperatura relativa del aire**
    - **ARIMA**

Algoritmo 27: Modelo ARIMA en R

```
fitARIMA <- arima(ts_m, order=c(6,1,1))
pred_arima <- forecast::forecast(fitARIMA, h = 24)

result_arima <- c(pred_arima[["x"]], pred_arima[["mean"]])
ts_arima <- ts(result_arima, start = c(2017, 1), frequency = 12)

plot(pred_arima, col="#284ECD" , fcol= "#D819E7", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", flwd="b", xaxt="none")

axis(1, at=1:length(index(ts_arima)), labels=index(ts_arima))
legend("topleft", legend=c("Real", "Predicción"),
       col=c("#284ECD", "#D819E7"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 74: Modelo ARIMA en R



Fuente: Elaboración propia

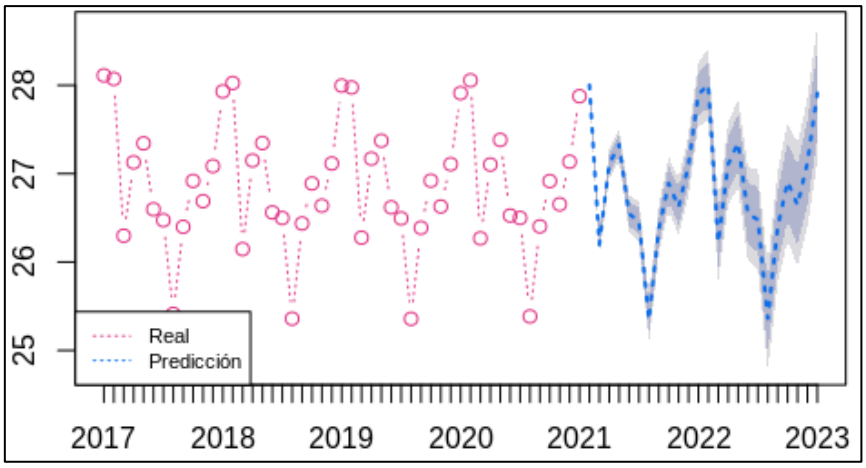
    - **SARIMA**

Algoritmo 28: Modelo SARIMA en R

```
modelo_sarima = Arima(ts_m, order=c(1,2,1), seasonal=list(order=c(1,1,1), period=12))
pred_sarima <- forecast::forecast(modelo_sarima, h = 24)

result_sarima <- c(pred_sarima[["x"]], pred_sarima[["mean"]])
ts_sarima <- ts(result_sarima, start = c(2017, 1), frequency = 12)

plot(pred_sarima, col="#E23687" , fcol= "#0B6EEC", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", xaxt="none")

axis(1, at=1:length(index(ts_sarima)), labels=index(ts_sarima))
legend("bottomleft", legend=c("Real", "Predicción"),
       col=c("#E23687", "#0B6EEC"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 75: Modelo SARIMA en R



Fuente: Elaboración propia

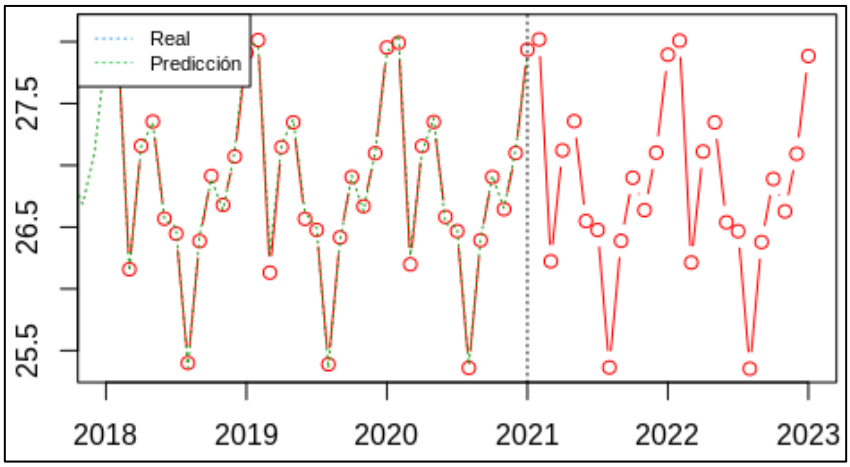
- **Holt-Winters**

Algoritmo 29: Modelo Holt-Winters en R

```
ts_hw <- ts(coredata(ts_m), start = c(2017, 1), frequency = 12)
hw <- HoltWinters(ts_hw)
pred_hw <- predict(hw, n.ahead = 24)

plot(hw, pred_hw, col="#109709", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor")
```

Fuente: Elaboración propia

Gráfico 76: Modelo Holt-Winters en R



Fuente: Elaboración propia

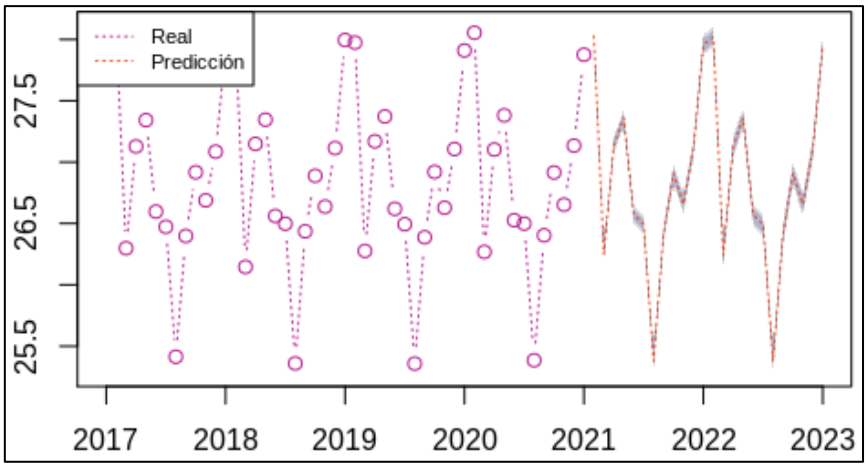
- **TBATS**

Algoritmo 30: Modelo TBATS en R

```
ts_tbats <- ts(coredata(ts_m), start = c(2017, 1), frequency = 12)

tbats_model = tbats(ts_tbats, seasonal.periods=c(12))
pred_tbats <- forecast(tbats_model, h=24)

result_tbats <- c(pred_tbats[["x"]], pred_tbats[["mean"]])
ts_tbats <- ts(result_tbats, start = c(2017, 1), frequency = 12)

plot(pred_tbats, col="#B81F93" , fcol= "#EE3D0D", type="b",
     main="Predicción sensor temperatura del aire", lty="dotted", xlab="Año",
     ylab="Promedio lectura sensor", flty="dotted", flwd="b")

axis(1, at=1:length(index(ts_tbats)), labels=index(ts_tbats))
legend("topleft", legend=c("Real", "Predicción"),
       col=c("#B81F93", "#EE3D0D"), lty=3:3, cex=.7)
```

Fuente: Elaboración propia

Gráfico 77: Modelo TBATS en R



Fuente: Elaboración propia

Anexo 3: Evidencia de pruebas de Escenario 1 de evaluación

- **Captura de datos en streaming**

Tabla 14: Evidencia pruebas T1, T2 y T3

Prueba	Simulador	Suscriptor MQTT	T1	Publicador Kafka	Consumidor Kafka	T2	Consumidor Kafka	MongoDB	T3	Total
1	2021-3-8 10:57:56.961	2021-4-8 10:57:56.963	**2**	2021-4-8 10:57:56.964	2021-4-8 10:57:56.982	**18**	2021-4-8 10:57:56.982	2021-4-8 10:57:56.983	**1**	
2	2021-3-8 10:57:57.240	2021-4-8 10:57:57.242	**2**	2021-4-8 10:57:57.245	2021-4-8 10:57:57.259	**14**	2021-4-8 10:57:57.259	2021-4-8 10:57:57.262	**3**	
3	2021-3-8 10:57:57.544	2021-4-8 10:57:57.549	**5**	2021-4-8 10:57:57.551	2021-4-8 10:57:57.560	**9**	2021-4-8 10:57:57.560	2021-4-8 10:57:57.563	**3**	
4	2021-3-8 10:57:57.861	2021-4-8 10:57:57.870	**9**	2021-4-8 10:57:57.878	2021-4-8 10:57:57.912	**34**	2021-4-8 10:57:57.912	2021-4-8 10:57:57.915	**3**	
5	2021-3-8 10:57:58.155	2021-4-8 10:57:58.160	**5**	2021-4-8 10:57:58.167	2021-4-8 10:57:58.192	**25**	2021-4-8 10:57:58.192	2021-4-8 10:57:58.197	**5**	
6	2021-3-8 10:57:58.456	2021-4-8 10:57:58.460	**4**	2021-4-8 10:57:58.468	2021-4-8 10:57:58.494	**26**	2021-4-8 10:57:58.494	2021-4-8 10:57:58.498	**4**	
7	2021-3-8 10:57:58.756	2021-4-8 10:57:58.762	**6**	2021-4-8 10:57:58.769	2021-4-8 10:57:58.792	**23**	2021-4-8 10:57:58.792	2021-4-8 10:57:58.796	**4**	
8	2021-3-8 10:57:59.062	2021-4-8 10:57:59.068	**6**	2021-4-8 10:57:59.081	2021-4-8 10:57:59.116	**35**	2021-4-8 10:57:59.116	2021-4-8 10:57:59.118	**2**	
9	2021-3-8 10:57:59.387	2021-4-8 10:57:59.403	**16**	2021-4-8 10:57:59.410	2021-4-8 10:57:59.423	**13**	2021-4-8 10:57:59.423	2021-4-8 10:57:59.425	**2**	
10	2021-3-8 10:57:59.688	2021-4-8 10:57:59.694	**6**	2021-4-8 10:57:59.701	2021-4-8 10:57:59.744	**43**	2021-4-8 10:57:59.744	2021-4-8 10:57:59.747	**3**	
Min			**2**			**9**			**1**	**12**
Prom			**6,1**			**24**			**3**	**33,1**
Max			**16**			**43**			**5**	**64**

Fuente: Elaboración propia

- **Integración y virtualización de datos**

Tabla 15: Evidencia pruebas T4 y T5

N° Obs	T4	T5	Total	Inicio	Fin		Inicio	Fin		Inicio	Fin		Inicio	Fin		Peso (MB)
				Anfitriona			Master			Anfitriona			Master			
				CPU (%)						RAM (GB)						
2524614	09:03.266	01:05.549		5 - 8	40 - 65	**35 - 57**	3 - 8	30 - 70	**27 - 62**	21,60	23,20	**1,60**	4,05	10,20	**6,15**	523,4
	08:56.091	01:00.304								21,60	23,10	**1,50**	4,02	10,00	**5,98**	
	08:56.491	00:59.129								21,30	22,90	**1,60**	4,04	10,00	**5,96**	
	09:00.134	00:56.734								21,40	23,10	**1,70**	3,99	10,10	**6,11**	
	09:02.146	00:58.348								21,10	22,80	**1,70**	4,02	10,00	**5,98**	
	08:58.468	01:01.482								21,40	23,00	**1,60**	4,03	9,99	**5,96**	
	09:00.587	00:57.942								21,50	23,20	**1,70**	4,03	10,20	**6,17**	
	08:56.367	00:59.127								21,30	23,10	**1,80**	4,04	10,00	**5,96**	
	09:02.495	01:00.367								21,30	22,90	**1,60**	4,02	10,10	**6,08**	
	08:57.671	00:58.648								21,50	23,10	**1,60**	3,96	10,00	**6,04**	
Min	**08:56.091**	**00:56.734**	**09:52.825**									**1,50**			**5,96**	
Prom	**08:59.372**	**00:59.763**	**09:59.135**									**1,64**			**6,04**	
Max	**09:03.266**	**01:05.549**	**10:08.815**									**1,80**			**6,17**	
1500000	05:28.435	00:30.925		5 - 7	35 - 56	**30 - 49**	3 - 10	34 - 55	**31 - 45**	21,50	23,30	**1,80**	4,03	7,47	**3,44**	295,8
	05:35.383	00:38.194								21,90	23,20	**1,30**	4,05	7,50	**3,45**	
	05:26.476	00:39.921								21,60	23,20	**1,60**	4,01	7,52	**3,51**	
	05:32.361	00:35.602								21,70	23,40	**1,70**	3,97	7,51	**3,54**	
	05:29.942	00:32.469								21,50	23,10	**1,60**	3,96	7,39	**3,43**	
	05:33.517	00:33.562								21,80	23,20	**1,40**	4,03	7,53	**3,50**	
	05:34.648	00:37.168								21,70	23,20	**1,50**	4,03	7,41	**3,38**	
	05:31.842	00:36.972								21,80	23,20	**1,40**	4,02	7,52	**3,50**	
	05:29.428	00:34.357								21,60	23,40	**1,80**	4,01	7,36	**3,35**	

	05:31.402	00:38.241								21,70	23,30	**1,60**	3,98	7,50	**3,52**	
Min	**05:26.476**	**00:30.925**	**05:57.401**									**1,30**			**3,35**	
Prom	**05:31.343**	**00:35.741**	**06:07.084**									**1,57**			**3,46**	
Max	**05:35.383**	**00:39.921**	**06:15.304**									**1,80**			**3,54**	
500000	02:33.179	00:10.402		6 - 8 %	35 - 52 %	**29 - 44**	3 - 8	36 - 49	**33 - 41**	21,60	23,20	**1,60**	4,05	5,47	**1,42**	102
	02:51.524	00:12.773								21,80	23,30	**1,50**	4,09	5,48	**1,39**	
	02:46.640	00:12.338								21,60	23,30	**1,70**	4,04	5,56	**1,52**	
	02:39.490	00:12.005								21,90	23,20	**1,30**	4,05	5,51	**1,46**	
	02:35.842	00:12.587								21,70	23,30	**1,60**	4,06	5,50	**1,44**	
	02:48.426	00:11.063								21,90	23,20	**1,30**	4,04	5,54	**1,50**	
	02:33.719	00:10.489								21,50	23,30	**1,80**	4,05	5,51	**1,46**	
	02:42.318	00:10.450								21,70	23,30	**1,60**	4,08	5,52	**1,44**	
	02:36.497	00:10.811								21,70	23,30	**1,60**	4,07	5,43	**1,36**	
	02:34.411	00:10.765								21,60	23,30	**1,70**	4,05	5,58	**1,53**	
Min	**02:33.179**	**00:10.402**	**02:43.581**									**1,30**			**1,36**	
Prom	**02:40.205**	**00:11.368**	**02:51.573**									**1,57**			**1,45**	
Max	**02:51.524**	**00:12.773**	**03:04.297**									**1,80**			**1,53**	

Fuente: Elaboración propia