BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

“SIMULADOR DE UN ORQUESTADOR DE SERVIDORES Y SERVICIOS EN LA NUBE PARA SAP”

TESIS

QUE PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

JOSE RODOLFO ROSAS LEZAMA

DIRECTOR DE TESIS

DR. MARIO ROSSAINZ LÓPEZ

No. de CVU: 928584

DICIEMBRE 2025

Agradecimientos

Al llegar al final de este camino, me siento profundamente agradecido con las personas que, con su amor, paciencia y apoyo, han sido fundamentales en mi proceso de formación y en la culminación de este proyecto.

En primer lugar, quiero agradecer de todo corazón a mi esposa, quien ha sido mi mayor fuente de fortaleza y motivación a lo largo de estos años. Desde el inicio de mi carrera, has estado a mi lado, apoyándome en cada paso del camino, entendiendo las largas horas de estudio, los momentos de estrés y la dedicación plena a mis proyectos. Gracias por brindarme tu comprensión, por ser mi refugio y mi aliento en los días más difíciles. Tu apoyo ha sido fundamental no solo para completar esta tesis, sino también para mantenerme firme en la búsqueda de mis sueños. Lo que hemos vivido juntos durante este tiempo es invaluable, y este logro es tan tuyo como mío.

A mis padres, mis más sinceros agradecimientos. Ustedes me enseñaron desde pequeño el valor del esfuerzo, la disciplina y la resiliencia ante la adversidad. En momentos de dificultad, cuando los retos parecían insuperables, su ejemplo de fortaleza me impulsó a seguir adelante. Gracias por mostrarme que los obstáculos no son barreras, sino oportunidades para crecer, y por ser siempre mi mayor fuente de inspiración. Su amor incondicional y su sacrificio son el motor que me ha permitido alcanzar esta meta. Sin su apoyo constante, este logro no hubiera sido posible.

A mis hermanos, gracias por ser mis primeros amigos y por brindarme su apoyo y compañía en cada etapa de mi vida. Su comprensión, sus consejos y, sobre todo, su cariño, han sido fundamentales para mantenerme equilibrado y motivado durante estos años. A pesar de las distancias y los momentos difíciles, siempre supe que podía contar con ustedes, y eso ha sido invaluable para mí. Este logro también es de ustedes, porque cada uno de ustedes ha sido parte de mi proceso de crecimiento.

A mis amigos, que han estado presentes en cada momento del viaje, gracias por su apoyo constante y por ofrecerme sus puntos de vista y consejos que siempre me ayudaron a ver las cosas desde nuevas perspectivas. En los días de incertidumbre, sus palabras fueron un recordatorio de que no estaba solo. Gracias por las horas de estudio compartido, por las risas, por los momentos de distracción y, sobre todo, por ser ese refugio donde pude encontrar comprensión y solidaridad. Ustedes son un pilar fundamental en este logro, y este éxito también lleva su huella.

Finalmente, a todas las personas que, de alguna u otra manera, han sido parte de este proceso, mi más sincero agradecimiento. Cada palabra de aliento, cada gesto de apoyo y cada momento compartido contribuyó a que hoy pueda celebrar este logro. Este trabajo es el resultado del esfuerzo colectivo de quienes, con su amor y dedicación, han hecho que este proyecto fuera posible.

Índice

1. Introducción

Actualmente las empresas están prestando mucha más atención a la optimización de procesos y recursos que utilizan para desarrollar sus diversos productos o servicios. Es por esto que en muchas de ellas el uso de un sistema de ERP (Enterprise Resource Planning) que es un sistema de software que se centra en la planificación de recursos los ayuda a manejar diferentes áreas de su negocio, desde finanzas hasta cadena de suministro y más. Este tipo de sistema integra en una sola plataforma los diversos procesos y funciones organizacionales permitiendo a sus clientes realizar la toma de decisiones de forma más rápida y eficiente, lo que en la industria significa agilizar procesos sin descuidar la calidad de sus productos, permitiéndoles generar beneficios no solo a las empresas si no al entorno que las rodea.

Cada uno de estos sistemas ERP utilizan diferentes tipos de hardware dependiendo de la cantidad de transacciones que una empresa realiza dentro de dichos sistemas, lo que va directamente relacionado con el tamaño y corte de cada una de ellas. Algunas empresas podrían utilizar solo un par de módulos conectados a un servidor, mientras que otras podrían utilizar todos los módulos disponibles dentro de un ERP ejecutándose de forma paralela en varios servidores. Dentro de los sistemas ERP existen diferentes propuestas, dentro de las que destacan, SAP, Oracle ERP Cloud, Microsoft Dynamics 365 o Infor. Cada una de ellas ofreciendo diferentes ventajas y costos, por lo que cada empresa puede escoger entre ellas priorizando la adaptabilidad de los sistemas a sus negocios. A su vez los sistemas de ERP se pueden implementar de forma local montando el sistema dentro de un servidor ubicado dentro de las diferentes empresas o desplegando en servidores virtuales montados en la nube de algún proveedor de nube pública. Claro está que los costos asociados a los diferentes tipos de instalaciones de hardware para desplegar los sistemas ERP varían dependiendo de si se realizan sobre servidores físicos o virtuales además del sistema ERP escogido y las versiones que se requieran instalar.

Este trabajo de tesis se centra en las empresas que utilizan sistemas ERP SAP en la nube de Amazon Web Services (AWS) las cuales enfrentan retos operativos relacionados con la administración de sus servidores virtuales, muy particularmente la necesidad de iniciar y apagar dichos servidores de manera eficiente. Esto derivado de las necesidades que hemos identificado al saber los costos por hora asociados a la renta de servidores virtuales que algunas veces no se utilizan debido a los tiempos no laborales de las diferentes empresas, algunas de ellas únicamente manejan horarios de lunes a viernes de 9 am a 5 pm, por lo que no tiene sentido mantener encendidos servidores virtuales que se sabe que no se utilizarán. Los administradores de sistemas requieren de herramientas que faciliten la gestión remota de estos servidores, asegurando un uso eficiente de los recursos mientras reducen costos operativos. El problema específico en este contexto es cómo realizar estas operaciones de forma eficiente y segura, considerando la arquitectura del ERP SAP, que en muchas ocasiones requiere de la configuración y control de múltiples instancias de máquinas virtuales que se encargan de distintos componentes del sistema ERP.

Es por esto que el presente trabajo plantea el diseño e implementación de una plataforma que permita llevar a cabo la administración de una lista de servidores virtuales montados en la nube de AWS que se utilicen junto a sistemas ERP SAP, los cuales requieren un manejo

extremadamente minucioso en las operaciones que desempeñan además de que cada uno de los procesos se debe llevar a cabo con los parámetros más altos de seguridad dado que cada uno de dichos servidores contiene y procesa información sensible relacionada con cada una de las empresas de los diferentes clientes, por lo que no es una opción el permitir algún tipo de acceso no autorizado.

1.1. Objetivo General

Diseñar e implementar un sistema de orquestación centralizada, utilizando una plataforma web segura para automatizar el ciclo de vida (encendido/apagado) de instancias EC2 que simulan servidores de producción SAP, con el fin de cuantificar la reducción de costos operativos de la infraestructura en la nube de AWS, proyectando una optimización en los tiempos no laborales.

1.2. Objetivos Particulares

- Diseñar el Modelo de Datos y la Arquitectura Lógica del Orquestador utilizando el estándar UML (Diagramas de Clases y Secuencia) para la gestión de la metadata de clientes y servidores.

- Implementar el Módulo de Seguridad (Autenticación y Autorización basada en Roles/RBAC) con un esquema de 2FA y JWT, logrando una disponibilidad del 99.9% en el acceso.

- Desarrollar y validar el Algoritmo de Orquestación (rutinas Start/Stop) para instancias EC2, garantizando un Tiempo Promedio de Ejecución (TPE) menor a 30 segundos por operación.

1.3. Limitaciones

- Se trabajó con servidores virtuales alojados en la nube pública de Amazon Web Services (AWS).
- Se utilizaron instancias EC2 de AWS para la simulación de los servidores de producción.
- Los sistemas Operativos con los que se trabajó fueron:
    - Linux Ubuntu, Windows Server 2019 y 2022 dentro de los servidores de producción.
    - Servidor Windows Server 2019 Datacenter para el servidor central (Bastión).

2. Marco Teórico

2.1. Sistemas ERP

Un ERP (Enterprise Resource Planning o Planificación de Recursos Empresariales) es un tipo de software diseñado para gestionar y automatizar los procesos centrales de una organización. Integra las diversas funciones de una empresa en un sistema único que permite a todos los departamentos acceder y compartir datos en tiempo real. Estos sistemas cubren áreas críticas como la contabilidad, recursos humanos, inventarios, logística, compras, ventas, manufactura, y otras, todo dentro de una plataforma.
Los primeros sistemas ERP tienen sus raíces en los sistemas de Planificación de Requerimientos de Materiales o MRP (Material Requirements Planning), los cuales surgieron en la década de 1960 y se centraban principalmente en la gestión de inventarios y la programación de la producción. Con el tiempo, la tecnología avanzó y los sistemas MRP evolucionaron hacia lo que hoy conocemos como ERP. En la década de los 90, empresas como SAP, Oracle y PeopleSoft comenzaron a dominar el mercado de ERP, durante esos años todas las instalaciones de sistemas ERP se llevaban a cabo de forma local (instalados en los servidores de la empresa), lo que requería grandes inversiones iniciales y recursos tecnológicos. Después en los años 2000, los sistemas ERP comenzaron a migrar hacia la nube, lo que permitió a las empresas reducir costos de infraestructura, mejorar la accesibilidad y escalar sus operaciones de manera más eficiente [36,37].

Componentes Principales de un ERP:
Un sistema ERP está compuesto por varios módulos interconectados, cada uno diseñado para gestionar un área específica de la empresa. Entre los módulos más comunes se encuentran:

- **Finanzas**: Gestiona la contabilidad, la planificación financiera, la facturación y los informes fiscales.
- **Ventas y CRM (Gestión de Relaciones con Clientes)**: Administra la gestión de pedidos, relaciones con clientes, gestión de oportunidades y análisis de ventas.
- **Logística y Gestión de Inventarios**: Controla la cadena de suministro, el almacenamiento y la distribución de productos.
- **Recursos Humanos**: Administra el reclutamiento, la nómina, los beneficios, el rendimiento de los empleados y la formación.
- **Producción y Manufactura**: Planifica y controla los procesos de fabricación, la gestión de la producción y los recursos necesarios para las operaciones.
- **Compras**: Gestiona el aprovisionamiento de bienes y servicios, controlando las relaciones con proveedores y la adquisición de materiales.
- **Gestión de Proyectos**: Facilita el seguimiento y la planificación de proyectos, incluyendo la asignación de recursos y presupuestos.

La característica clave de los sistemas ERP es que estos módulos están integrados, lo que significa que los datos generados por un departamento están disponibles de forma inmediata para otros departamentos, mejorando la comunicación organizacional.

Características de los Sistemas ERP:
Algunas de las principales características de los sistemas ERP son:

- **Automatización de procesos**: Permiten automatizar una gran cantidad de tareas repetitivas y manuales, lo que reduce el riesgo de errores humanos y mejora la eficiencia operativa.
- **Mejora en la toma de decisiones**: Al integrar los datos de todos los departamentos en tiempo real, permiten tener una visión precisa del estado de la empresa en todo momento.
- **Optimización de recursos**: Ayudan a las empresas a gestionar de manera más eficiente sus recursos, como el tiempo, los materiales, la mano de obra y el dinero, lo que puede llevar a una reducción de costos.
- **Escalabilidad y flexibilidad**: Además, los sistemas ERP basados en la nube permiten escalar a medida que la empresa crece, lo que permite agregar nuevos módulos y funcionalidades minimizando la inversión en infraestructura tecnológica.

Principales Proveedores de ERP:
Existen diversos proveedores de soluciones ERP en el mercado, siendo algunos de los más conocidos:

- **SAP**: Es una de las plataformas más utilizadas en grandes organizaciones.
- **Oracle ERP Cloud**: Popular en grandes empresas, ofrece soluciones en la nube.
- **Microsoft Dynamics 365**: Plataforma que integra ERP y CRM, orientada a pequeñas y medianas empresas.
- **Infor**: Ofrece soluciones ERP con enfoque en industrias específicas.

En esta tesis se aborda el uso del sistema ERP SAP (Systems, Applications and Products in Data Processing), que es uno de los principales sistemas de planificación de recursos empresariales (ERP) utilizados por las organizaciones para gestionar procesos de negocio.

2.2. ERP SAP

La compañía Systems, Applications, and Products in Data Processing (SAP) es una de las empresas líderes a nivel mundial en el desarrollo de software empresarial. Se fundó en 1972 en Alemania y con el paso del tiempo se ha consolidado como el proveedor más reconocido de sistemas ERP, ofreciendo a las organizaciones soluciones que permiten eficientar los procesos dentro de sus negocios [38].

Algunos de los productos ofrecidos por SAP son:

- **SAP ERP Central Component (ECC)**: La solución ERP tradicional que incluye módulos como finanzas, gestión de inventarios, compras, recursos humanos y manufactura.
- **SAP S/4HANA**: Una suite ERP de última generación basada en la tecnología in-memory de HANA, que proporciona análisis en tiempo real y mejor rendimiento.
- **SAP Business One**: Diseñada para pequeñas y medianas empresas, ofrece funcionalidad ERP con menor costo y complejidad.

- **SAP Ariba**: Enfocada en la gestión de adquisiciones y la cadena de suministro.
- **SAP SuccessFactors**: Herramienta para la gestión del talento y recursos humanos.
- **SAP Concur**: Especializada en la gestión de gastos corporativos y viajes.

Los módulos principales de SAP son:
- **Gestión Financiera**: Contabilidad, gestión de tesorería y control presupuestario.
- **Gestión de Recursos Humanos**: Nómina, reclutamiento y desarrollo del talento.
- **Gestión de la Cadena de Suministro**: Planificación de la demanda, compras y gestión de inventarios.
- **Ventas y Marketing**: CRM y gestión de órdenes de clientes.
- **Producción y Logística**: Planificación y control de manufactura.

SAP utiliza una arquitectura de tipo cliente-servidor distribuida que consta principalmente de tres capas:
- **Capa de presentación**: Interfaz de usuario (usualmente un cliente SAP GUI o navegador web).
- **Capa de aplicación**: Servidores de aplicación que procesan la lógica de negocio.
- **Capa de base de datos**: Servidores de base de datos que almacenan toda la información relacionada con los procesos empresariales.

En el entorno SAP, los perfiles de usuario se estructuran en función de sus roles y responsabilidades específicas, que varían según el ambiente del sistema (Desarrollo, Calidad o Productivo) en el que interactúan. Entre los principales tipos de perfiles de usuario se encuentran los siguientes:

Perfiles Técnicos y de Consultoría
Estos perfiles son responsables de la configuración, el desarrollo y el mantenimiento técnico del sistema [49].
- **Consultor Funcional (Functional Consultant):**
  Es un experto en módulos específicos de SAP (como Finanzas (FI), Gestión de Materiales (MM), Ventas y Distribución (SD), etc.). Trabaja en estrecha colaboración con los usuarios de negocio para recopilar requisitos y traducir esos requisitos en configuraciones del sistema en el ambiente DEV. Su rol implica la parametrización del sistema (customizing) y, a menudo, la coordinación de las pruebas de integración en QAS.
- **Desarrollador ABAP (ABAP Developer):**
  Se enfoca en la parte técnica, creando, modificando o mejorando programas utilizando ABAP, el lenguaje de programación nativo de SAP. Estos usuarios trabajan predominantemente en el ambiente DEV, donde desarrollan nuevas funcionalidades, interfaces e informes personalizados, y realizan pruebas unitarias iniciales.
- **Administrador SAP Basis:**
  Este perfil es responsable de la infraestructura técnica subyacente del sistema SAP. Sus responsabilidades incluyen la instalación, configuración, gestión del rendimiento, administración de bases de datos, y la gestión de usuarios y accesos en todos los ambientes (DEV, QAS, Sandbox y PROD). Se aseguran de que el flujo de órdenes de transporte entre ambientes funcione correctamente.

- **Consultor de Seguridad y GRC (Governance, Risk, and Compliance):**
  Define y mantiene la arquitectura de roles y perfiles de seguridad en todo el panorama de sistemas. Este rol es crucial para asignar las autorizaciones correctas a los usuarios y garantizar la auditoría y el cumplimiento normativo en todos los ambientes.

Tipos de Ambientes SAP:
La arquitectura de un cliente SAP típicamente sigue un modelo de tres niveles (3-Tier Landscape) o más. Dentro de los principales ambientes en la arquitectura SAP se encuentran los siguientes:

- **Ambiente de Desarrollo (DEV - Development):**
  En este ambiente es donde los consultores y desarrolladores configuran, personalizan y programan nuevas funcionalidades, reportes, interfaces y extensiones (como por ejemplo, el desarrollo ABAP). Es un ambiente de pruebas iniciales y unitarias. Su propósito principal consiste en construir y personalizar el sistema SAP de acuerdo con los requisitos del negocio. Los cambios realizados aquí se guardaban en "órdenes de transporte" (transport requests) y, una vez completadas las pruebas unitarias iniciales, se movían al siguiente ambiente.

- **Ambiente de Calidad (QAS - Quality Assurance / Aseguramiento de Calidad):**
  Una vez que los cambios se desarrollan y prueban unitariamente en DEV, se transportan a QAS. En este ambiente, los usuarios clave y funcionales realizan pruebas más exhaustivas, conocidas como pruebas de integración y pruebas de aceptación del usuario (UAT). El objetivo es asegurar que los cambios funcionen correctamente y no interfieran con otras funciones del sistema.

- **Ambiente Sandbox (SBX):**
  Un Sandbox es un ambiente independiente, aislado y copiado del sistema productivo o con datos de prueba, que se usa para experimentación libre, formación, pruebas de concepto, o para replicar y solucionar errores específicos sin afectar los ambientes DEV o QAS. Los cambios hechos aquí no suelen seguir el flujo normal de transporte hacia producción.
  El ambiente de Garantía de Calidad se configuró para simular el entorno de producción tan fielmente como fuera posible. El objetivo de este sistema fue asegurar que todos los cambios desarrollados en DEV funcionaran correctamente y sin introducir errores en los procesos existentes (Community SAP, 2007). Dentro de las pruebas que se realizan en este perfil se encuentran:
    - Pruebas de integración (integration testing).
    - Pruebas de aceptación del usuario (UAT, por sus siglas en inglés).
    - Pruebas de regresión, para verificar que las nuevas funcionalidades no impactaran negativamente las existentes.

- **Producción (PRD):**
  Su propósito es dar soporte a las operaciones diarias del negocio (ventas, finanzas, logística, manufactura). En este el tiempo de inactividad genera pérdidas directas de

negocio (ingresos, multas, interrupción de procesos). Los ambientes anteriores a este, forman la base de una estrategia de gestión de cambios (Change Management) en SAP, asegurando que solo los cambios probados y aprobados lleguen al sistema Productivo (PROD), minimizando así el riesgo de interrupciones operativas.

Perfiles de Negocio
Estos perfiles representan a los usuarios finales y expertos en la materia que utilizan el sistema para las operaciones diarias o para validar cambios.

- **Usuario Clave (Key User) / Superusuario (Power User):**
  Posee un conocimiento profundo de los procesos de negocio en su área funcional y actúa como enlace entre los usuarios finales y el equipo de IT. Los Key Users tienen permisos avanzados y desempeñan un papel fundamental en la ejecución de las pruebas de aceptación del usuario (UAT) en el ambiente QAS, y a menudo brindan soporte de primer nivel a otros usuarios en el ambiente Productivo (PROD).
- **Usuario Final (End User) / Usuario de Diálogo (Dialog User):**
  Es el usuario regular que realiza tareas operativas diarias en el ambiente Productivo (PROD), como ingresar pedidos de venta, registrar facturas o generar informes. Por lo general, tienen acceso limitado y solo visualizan las transacciones y datos relevantes para su función específica. También pueden utilizar el ambiente Sandbox o QAS para fines de formación.

La infraestructura tecnológica que soporta a SAP está constituida por servidores que deben estar configurados para soportar cargas de trabajo intensivas y una alta disponibilidad. Los servidores de base de datos son responsables de almacenar grandes volúmenes de datos garantizando la rápida recuperación de la información. Los servidores de aplicaciones ejecutan la lógica de negocio de los módulos SAP y deben estar optimizados para soportar múltiples usuarios concurrentes.
Con tecnologías como VMware o Hyper-V permite a las empresas optimizar el uso de los recursos de hardware al permitir la ejecución de múltiples instancias de SAP en un único servidor físico. Según un estudio realizado por SAP (2021), la virtualización puede reducir hasta un 40% los costos operativos relacionados con la infraestructura del servidor, mejorando la escalabilidad y gestión de los recursos.
Los servidores que se utilizan en las instalaciones de los sistemas ERP SAP, deben proveer, cpu de alto rendimiento, memoria ram ya que SAP consume una gran cantidad de memoria para manejar las operaciones en tiempo real. Además de almacenamiento de alto rendimiento, los sistemas SAP requieren almacenamiento rápido y redundante (RAID) para garantizar la disponibilidad de datos. El rendimiento del sistema SAP está directamente relacionado a las capacidades del hardware utilizado.
A su vez, a medida que las empresas crecen, también lo hacen sus necesidades de procesamiento de datos, las soluciones de nube y virtualización proporcionan una respuesta flexible a la escalabilidad. Eso aunado a la correcta configuración, mantenimiento y optimización de los servidores permite maximizar los beneficios de SAP.

2.2.1. SAP Basis

Es un conjunto de herramientas que permite la administración de la arquitectura del sistema SAP con el objetivo de garantizar el buen funcionamiento de la infraestructura [8]. Esta herramienta permite:

- El monitoreo, ajuste de carga y el rendimiento de las aplicaciones.
- Mantenimiento a bases de datos, sistemas operativos, aplicaciones y servidores web que están en el panorama del sistema SAP.
- Detección e inicio de sistemas como tareas de mantenimiento, actualizaciones o cambios en la configuración.
- Administración de perfiles de usuario.
- Operación y gestión de errores implementando y resolviendo errores utilizando herramientas como SNOTE (SAP Note Assistant).

2.3. Servidores

Un servidor es un tipo de computadora o software diseñado para proporcionar servicios, recursos o datos a otros sistemas, computadoras o usuarios conocidos como clientes, a través de una red. Su función principal es recibir solicitudes de sus clientes y responder a esas solicitudes proporcionando los recursos o datos solicitados. Los servidores juegan un papel crucial en la infraestructura de redes y en la arquitectura de sistemas informáticos, facilitando la comunicación y el intercambio de información entre diferentes partes de un sistema [11, 12].

Tipos de Servidores:

- **Servidor Web**: Su función es almacenar y servir páginas web a los navegadores de los usuarios. Se usan para hospedar sitios y aplicaciones web, estos servidores los podemos encontrar en internet alojando la página de tiendas en línea, blogs o dentro de una red cerrada. Algunos servidores web son: Apache, Nginx y Microsoft IIS, etc.
- **Servidor de Archivos:** Permiten el almacenamiento y acceso a archivos compartidos a través de una red. Los usuarios pueden guardar, acceder y compartir archivos. Proporcionan almacenamiento centralizado para documentos, imágenes y otros archivos dentro de una organización.
- **Servidor de Correo Electrónico:** Manejan el envío, recepción y almacenamiento de correos electrónicos. Facilitan la comunicación por correo electrónico tanto a nivel personal como empresarial. Por ejemplo, Gmail, Microsoft Exchange, etc.
- **Servidor de Base de Datos:** Gestiona y almacena bases de datos, permitiendo el acceso y la manipulación de datos estructurados. Almacenan y gestionan grandes volúmenes de datos para aplicaciones, sitios web y sistemas de gestión empresarial. Por ejemplo, MySQL, PostgreSQL y Oracle Database.
- **Servidor de Aplicaciones:** Ejecutan aplicaciones empresariales, facilitando la interacción entre usuarios y aplicaciones. Proporcionan un entorno para ejecutar aplicaciones empresariales y servicios backend. Por ejemplo, IBM WebSphere y Red Hat JBoss.
- **Servidor de DNS (Domain Name System):** Nos permite encontrar las direcciones ip pertenecientes a un nombre de dominio. Cuando se ingresa un nombre de dominio en un navegador, el servidor DNS traduce este nombre en una dirección IP que puede ser entendida por la red. Facilita la navegación en la web al traducir nombres de dominio en direcciones IP.

- **Servidor Proxy:** Actúa como intermediario entre los clientes y otros servidores, proporcionando características como filtrado de contenido, almacenamiento en caché y anonimato. Mejora la seguridad, el rendimiento y la privacidad en la navegación web. Se podrían utilizar por ejemplo para acceder al contenido bloqueado geográficamente al redireccionar el tráfico del usuario a través de sus servicios.

Los servidores suelen trabajar en diferentes tipos de arreglos de red conocidos como modelos, por ejemplo, el modelo Cliente-Servidor en donde el servidor proporciona recursos o servicios y el cliente realiza solicitudes a esos servicios como en los tipos de servidores antes mencionados.
Por lo cual un servidor es una pieza fundamental de la infraestructura de red que proporciona servicios y recursos a otras computadoras. Puede ser físico o virtual, y puede desempeñar diversas funciones dependiendo del tipo de servidor y las necesidades de la red o de la organización.

Características Comunes de los Servidores:
Hardware: Los servidores suelen tener hardware más potente y especializado en comparación con las computadoras de escritorio, incluyendo más memoria RAM, procesadores más rápidos y almacenamiento en disco de mayor capacidad.

Sistema Operativo: Operan con sistemas operativos diseñados para gestionar cargas de trabajo pesadas y proporcionar servicios de red, como Windows Server, Linux y Unix.

Redundancia: Están diseñados para ser altamente fiables y funcionar de manera continua. A menudo cuentan con características como fuentes de alimentación redundantes y sistemas de respaldo para minimizar el tiempo de inactividad.

Seguridad y Administración: Los servidores requieren una gestión adecuada de seguridad, configuración y mantenimiento para proteger los datos y garantizar su disponibilidad.

2.3.1. Servidores en la nube

Un servidor en la nube es un recurso de computación virtualizado que se aloja y se gestiona en un entorno de nube. En lugar de estar físicamente ubicado en un centro de datos local o en una instalación propia, el servidor en la nube reside en una infraestructura de nube proporcionada por un proveedor de servicios en la nube [14].

Características de los servidores en la nube:
- **Virtualización:** Los servidores en la nube suelen ser máquinas virtuales (VM) que se ejecutan sobre una infraestructura física compartida. Esta virtualización permite a múltiples servidores virtuales coexistir en una sola máquina física, optimizando el uso de recursos.
- **Escalabilidad:** Los servidores en la nube pueden ser fácilmente escalados hacia arriba o hacia abajo según las necesidades. Esto significa que puedes aumentar o reducir la capacidad del servidor en función de la demanda, sin necesidad de adquirir hardware adicional.
- **Acceso Remoto:** Estos servidores se acceden y gestionan a través de internet, lo que proporciona flexibilidad para trabajar desde cualquier lugar.

- **Modelo de Pago por Uso:** En lugar de adquirir hardware y software, los usuarios pagan por el uso de los recursos en función del tiempo y la capacidad consumidos. Esto es más económico dado que solo pagas por lo que usas.
- **Administración y Mantenimiento:** El proveedor de servicios en la nube se encarga del mantenimiento de la infraestructura subyacente, como el hardware, las actualizaciones de software, y la seguridad. Lo que libera al usuario de las tareas de administración y mantenimiento.
- **Redundancia:** Los proveedores de servicios en la nube ofrecen redundancia y recuperación ante desastres para asegurar la disponibilidad continua y la integridad de los datos.

Proveedores de servicios en la nube:
- **Amazon Web Services (AWS):** Ofrece una amplia gama de servicios en la nube, incluyendo Amazon EC2 para instancias de servidores.
- **Microsoft Azure:** Proporciona máquinas virtuales y otros servicios de computación en la nube.
- **Google Cloud Platform (GCP):** Incluye Google Compute Engine para instancias de servidores virtuales.

La presente propuesta se centra en los servicios ofrecidos por la nube de AWS, dado que a pesar de que existen más proveedores de servicios en la nube, la mayoría de los clientes que han realizado la migración de sus sistemas SAP hacia la nube, lo han hecho hacia AWS dado las facilidades como flexibilidad y escalabilidad que permite a sus organizaciones, almacenando diferentes volúmenes de datos y permitiéndoles realizar cargas de trabajo variables.

2.4. AWS

Amazon Web Services (AWS) es una plataforma líder de servicios de computación en la nube, lanzada en 2006 por Amazon. AWS ofrece una infraestructura global escalable que permite a empresas y desarrolladores construir, implementar y gestionar aplicaciones de manera flexible y rentable.
AWS opera en un modelo de pago por uso, proporcionando servicios que abarcan almacenamiento, bases de datos, computación, redes, inteligencia artificial, análisis de datos, y más. Su infraestructura global está organizada en regiones (geográficas) y zonas de disponibilidad (data centers dentro de una región), lo que garantiza disponibilidad, resiliencia y bajas latencias.

AWS cuenta con más de 200 servicios disponibles, de los cuales los siguientes son los que están relacionados con el actual trabajo de tesis.
- **EC2 (Elastic Compute Cloud)**: Instancias de Computación
    - Ofrece capacidad de computación escalable en la nube, permitiendo a los usuarios lanzar servidores virtuales (instancias) según las necesidades específicas.
    - Soporta diferentes configuraciones (tipos de instancias) optimizadas para tareas como computación intensiva, memoria o almacenamiento.
    - Beneficios: Elasticidad, pago por segundo y soporte para entornos heterogéneos como Linux y Windows.

- **S3 (Simple Storage Service)**: Almacenamiento de Objetos

    - Servicio diseñado para almacenar y recuperar cualquier cantidad de datos desde cualquier lugar. Ideal para copias de seguridad, almacenamiento de archivos y análisis de big data.
    - Características principales: alta durabilidad (99.999999999%) y acceso seguro a través de políticas de permisos.

- **RDS (Relational Database Service)**: Bases de Datos Relacionales
    - Facilita la configuración, operación y escalado de bases de datos relacionales en la nube.
    - Soporta motores populares como MySQL, PostgreSQL, MariaDB, Oracle, SQL Server y Amazon Aurora.
    - Automatiza tareas como backups, actualizaciones y escalado de almacenamiento.

- **IAM (Identity and Access Management)**: Gestión de Identidades y Accesos
    - Proporciona control granular sobre quién puede acceder a los recursos de AWS y qué acciones pueden realizar.
    - Soporta la creación de roles, políticas y usuarios, promoviendo una gestión segura y eficiente.

- **CloudWatch**: Monitoreo y Observabilidad
    - Servicio diseñado para monitorear aplicaciones y recursos en AWS.
    - Permite recopilar métricas, generar alarmas y visualizar datos para garantizar la salud y el rendimiento de los sistemas.

- **AWS Lambda**: Computación sin Servidor (Serverless)
    - Ejecuta código en respuesta a eventos sin necesidad de gestionar servidores.
    - Útil para crear aplicaciones altamente escalables y pagar únicamente por el tiempo de ejecución utilizado.

- **VPC (Virtual Private Cloud)**: Redes Privadas
    - Permite crear redes virtuales aisladas dentro de AWS, otorgando control sobre el rango de IP, subredes, tablas de enrutamiento y puertas de enlace de acceso.

- **AWS Elastic Beanstalk**: Implementación Simplificada de Aplicaciones
    - Herramienta que facilita el despliegue y la administración de aplicaciones web desarrolladas en lenguajes como Java, Python, Node.js, entre otros.

- **DynamoDB**: Base de Datos NoSQL
    - Base de datos NoSQL completamente gestionada, optimizada para alto rendimiento y baja latencia. Ideal para aplicaciones que requieren escalabilidad automática.

- **CloudFront**: Red de Distribución de Contenido (CDN)
    - Servicio que acelera la entrega de contenido como videos, imágenes y datos dinámicos al usuario final, reduciendo latencias.

La tecnología de AWS nos brinda la herramienta AWS Systems Manager que permite la automatización de procesos para administrar y operar aplicaciones SAP.

Esta herramienta permite a los clientes administrar, actualizar y parchear servidores y máquinas virtuales (VM) en AWS, en las instalaciones y en otras nubes desde una única consola a escala [1].

También va a brindar la capacidad de personalizar los sistemas SAP con la ayuda de scripts de configuración previa y posterior. Los scripts de configuración previos a la implementación se ejecutan inmediatamente después de que se lanzan las instancias y se completan las actividades de configuración básicas del Asistente de lanzamiento, como la implementación de agentes de AWS CLI, Amazon CloudWatch y AWS Systems Manager.

SSM Application Manager permite específicamente comprender y descubrir la topología del panorama de SAP para los servidores de aplicaciones SAP NetWeaver ABAP y las bases de datos de SAP HANA, y sus interdependencias [7]. En la figura 1, podemos ver la estructura general de los servicios para una instancia de AWS EC2.

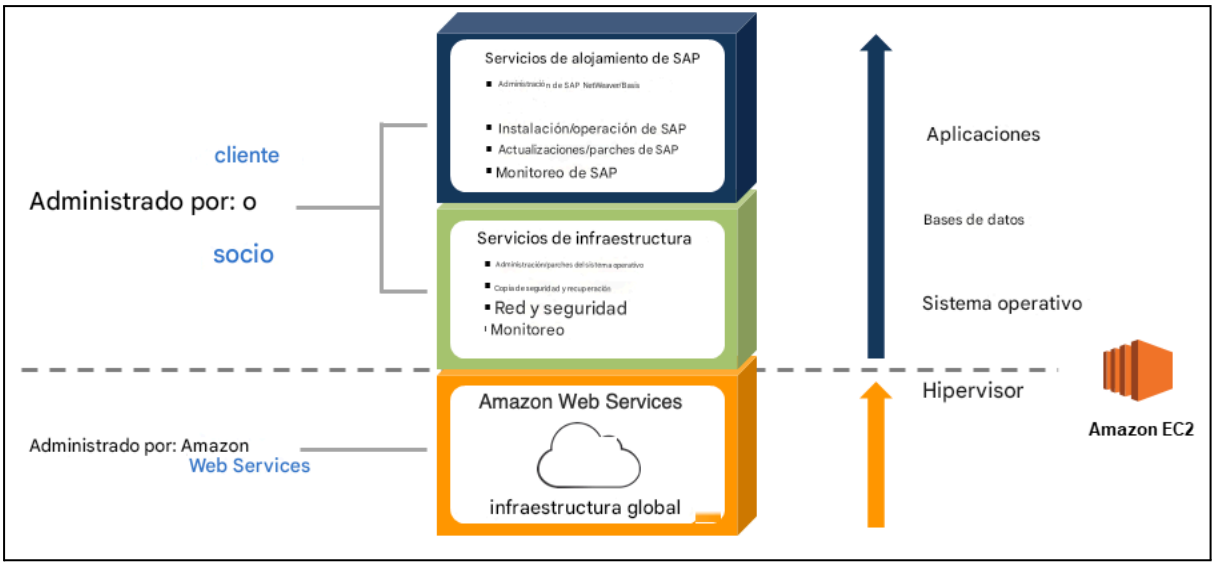


Fig 1. Servicios gestionados para SAP en AWS. Tomada de [49].

Al configurar un entorno SAP en AWS, deberá configurar un sistema SAP Solution Manager y un SAProuter con una conexión a la red de soporte de SAP, como lo haría con cualquier infraestructura. Como se muestra en la anterior figura 2.

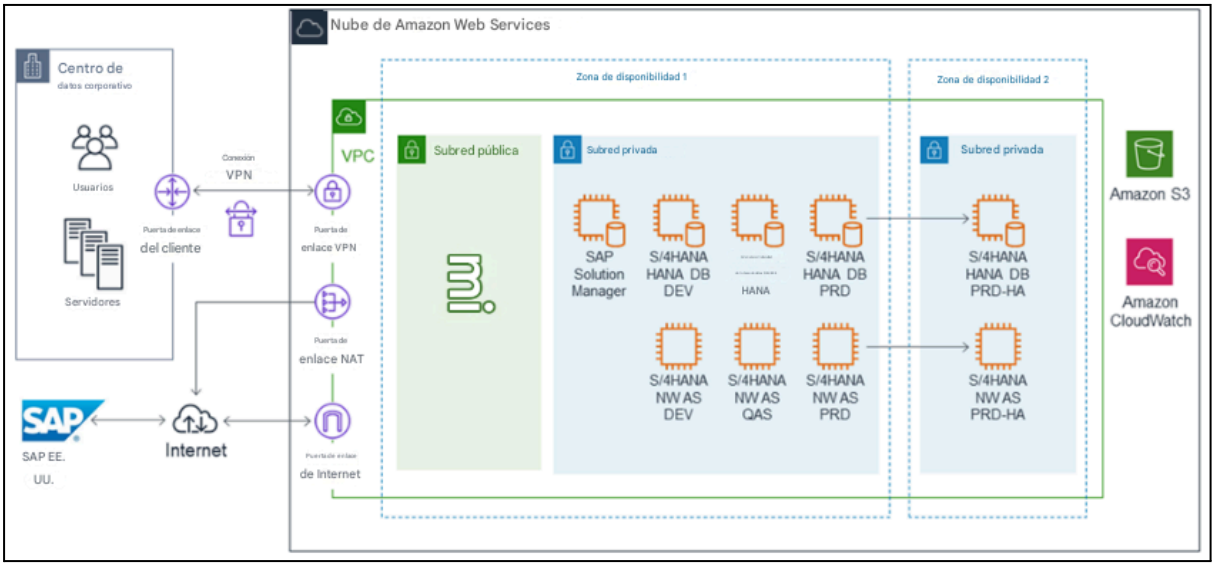


Fig 2. Arquitectura de SAP all-on-aws. Tomada de [49].

Pasos para la configuración del entorno SAP en AWS:
Iniciar la instancia en la que está instalado el software SAProuter en una subred pública de la VPC y asignar dirección IP elástica.
Crear un grupo de seguridad específico para la instancia de SAProuter con las reglas necesarias para permitir el acceso entrante y saliente requerido a la red de soporte de SAP.
Utilizar el tipo de conexión a Internet Secure Network Communications (SNC).

En la figura 3 podemos observar el despliegue del sistema SAP en una estructura híbrida, una parte de forma local y la otra parte en la nube mediante AWS.
En la industria también se pueden encontrar herramientas que nos permiten la administración de servidores que consisten en gestionar la infraestructura del servidor y supervisar su rendimiento mediante el monitoreo continuo.

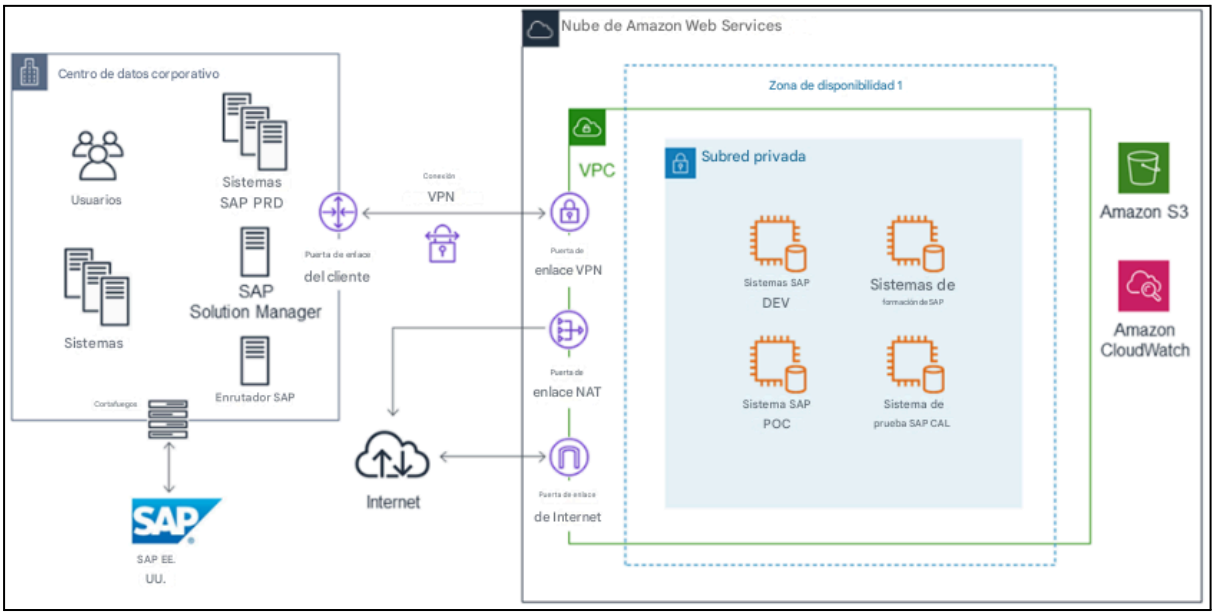


Fig 3. Arquitectura híbrida de SAP AWS. Tomada de [49].

2.5. Modelos de comunicación

La arquitectura de una red se refiere a la estructura y diseño de los componentes dentro de una red de comunicaciones. Existen varios tipos de arquitecturas de red, cada una adecuada para diferentes necesidades y contextos.

2.5.1. Cliente-Servidor

El modelo Cliente-Servidor es un patrón arquitectónico fundamental en redes y sistemas de software que organiza la comunicación y el procesamiento entre dos tipos de entidades: clientes y servidores. Este modelo es ampliamente utilizado en diversas aplicaciones y servicios, desde la web hasta bases de datos y sistemas distribuidos. En este modelo los clientes (computadoras, dispositivos u otros programas) solicitan servicios o recursos a un servidor centralizado, que los proporciona. Así, los servicios se encuentran centralizados por lo que los clientes no necesitan almacenar grandes cantidades de datos dado que los servidores manejan la carga y el mantenimiento centralizado de los mismos [13]. En la figura 4, podemos distinguir cada una de las partes que lo componen.

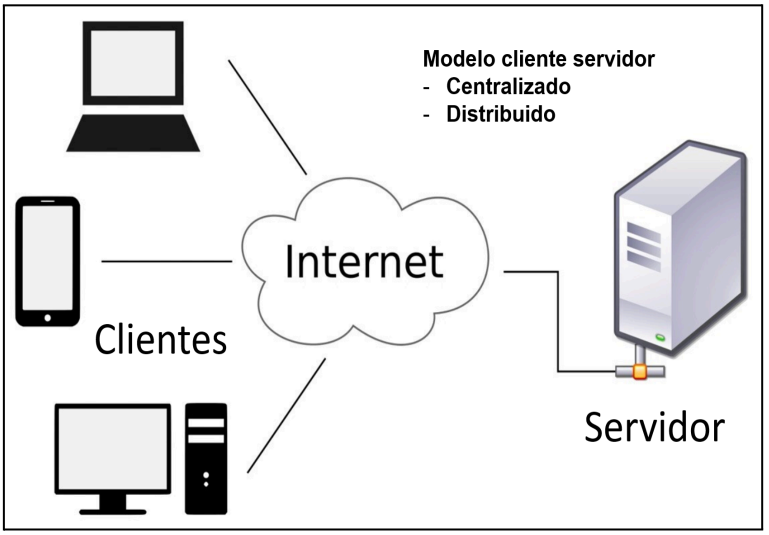


Fig 4. Diagrama del modelo Cliente-Servidor. Tomada de [50].

Componentes de este modelo:
Cliente: Es una entidad que solicita servicios o recursos al servidor, generalmente son aplicaciones o dispositivos que interactúan con el servidor para obtener datos o realizar operaciones.

Servidor: Es una entidad que proporciona servicios, recursos o datos a los clientes. Los servidores están diseñados para recibir y procesar las solicitudes de los clientes devolviendo las respuestas correspondientes.

Características principales:
Desacoplamiento: Los clientes y servidores tienen roles claramente definidos. Los clientes inician solicitudes, mientras que los servidores responden a las mismas.

Interacción: Los clientes y servidores no necesitan estar directamente conectados; la comunicación se realiza a través de redes o protocolos estandarizados.

Escalabilidad: Los servidores pueden ser escalados verticalmente (aumentando recursos en un solo servidor) u horizontalmente (añadiendo más servidores) para manejar un mayor número de solicitudes. Por su parte los clientes pueden ser múltiples y distribuidos, permitiendo una amplia gama de dispositivos y aplicaciones que interactúan con los servidores.

Centralización de Recursos: Los servidores centralizan el almacenamiento y el procesamiento de datos, lo que facilita la gestión y la administración de recursos. Esto permite un control más eficiente sobre la seguridad, las actualizaciones y el mantenimiento de los servicios y datos.

Ventajas:
Centralización: Facilita la administración centralizada de datos y recursos, lo que simplifica la gestión y el mantenimiento.
Escalabilidad: Permite escalar los recursos del servidor para manejar un mayor número de clientes o solicitudes.
Desacoplamiento: Permite que clientes y servidores evolucionen de manera independiente, facilitando la actualización y el mantenimiento.

Desventajas:
Punto Único de Fallo: Los servidores pueden convertirse en un punto único de fallo si no están diseñados para alta disponibilidad.
Carga del Servidor: El servidor puede verse sobrecargado si no está dimensionado adecuadamente para manejar la cantidad de solicitudes de los clientes.
Seguridad: La centralización de recursos puede representar un riesgo si no se implementan adecuadamente las medidas de seguridad.

2.5.2. Maestro-Esclavo

Esta es una arquitectura de comunicación en redes y sistemas donde un componente, denominado "Maestro", controla o coordina las operaciones de uno o varios componentes denominados "Esclavos" [13].

Componentes de este modelo:
Maestro: Este componente controla la operación de los esclavos. Puede enviar comandos, solicitar datos y coordinar las acciones.

Esclavo: Estos componentes responden a las solicitudes del maestro. Ejecutan comandos, envían datos al maestro y actúan según las instrucciones recibidas.

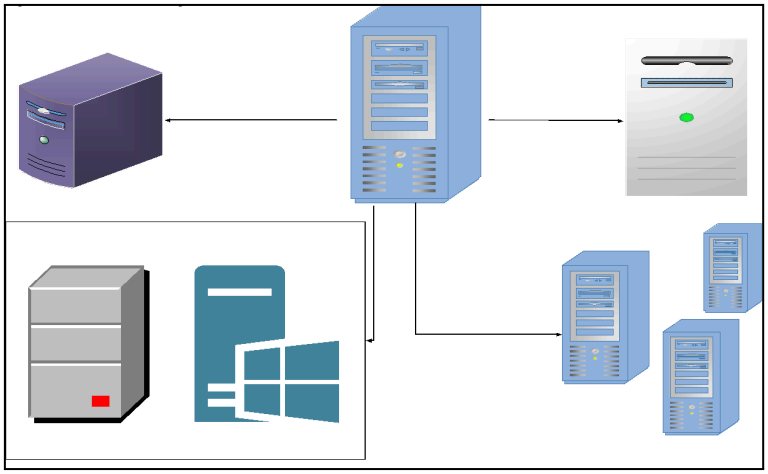


Fig 5. Diagrama del modelo Maestro-Esclavo. Diseño propio.

Comunicación Unidireccional: La comunicación se realiza desde el maestro hacia los esclavos. Lo que significa que los esclavos únicamente responden a las solicitudes del maestro.
Sincronización: El maestro a menudo coordina y sincroniza las actividades de los esclavos. Esto puede incluir la programación de tareas, la sincronización de datos o la gestión de recursos compartidos.
Jerarquía: El maestro tiene autoridad sobre los esclavos. Esto significa que el maestro tiene el control, mientras que los esclavos actúan en función de las órdenes recibidas. En la figura 5, podemos distinguir cada una de las partes que lo componen.

Ejemplos de este modelo de comunicación:
Redes y Comunicaciones: Algunos protocolos de comunicación, como el I2C (Inter-Integrated Circuit), utilizan un modelo maestro-esclavo. En I2C, un dispositivo maestro controla el bus y coordina la comunicación con varios dispositivos esclavos.

Bases de Datos y Sistemas de Almacenamiento: En este entorno un servidor maestro puede gestionar y coordinar varias réplicas de bases de datos esclavas para la distribución de datos y redundancia.

Ventajas:
Simplicidad: La estructura jerárquica y la comunicación unidireccional simplifican el diseño y la implementación.
Control Centralizado: El maestro tiene un control centralizado sobre la operación de los esclavos, lo que facilita la coordinación y la gestión.

Desventajas:
Punto Único de Fallo: Si el maestro falla, toda la operación puede verse afectada, ya que los esclavos dependen del maestro para recibir comandos.
Escalabilidad: El modelo puede tener limitaciones de escalabilidad, ya que un número excesivo de esclavos puede sobrecargar al maestro.
Este modelo es una arquitectura efectiva para la coordinación y control de sistemas distribuidos, proporcionando una estructura clara para la gestión y comunicación.

Este modelo de comunicación es importante dado que será el modelo que utilizaremos en la operación de cada uno de los servidores de producción de cada uno de los clientes (siendo estos servidores los esclavos). El maestro será el servidor bastión el cual contendrá el

registro de todos los clientes, los negocios, la dirección de los servidores y las claves asociadas a cada uno de ellos. Con esto en mente, el servidor bastión ordenará el encendido o apagado remoto de algún servidor esclavo y este deberá de seguir las instrucciones recibidas.

2.5.3. Tipos de redes de comunicación

Red de Área Local (LAN): Una LAN conecta dispositivos dentro de un área geográfica limitada, como una oficina o un edificio.
Este tipo de red proporciona alta velocidad de transmisión de datos, son redes de propiedad privada o corporativa. Usualmente utilizan tecnologías como Ethernet o Wi-Fi. Se menciona el presente modelo dado que la mayoría de los clientes que trabajan con el sistema SAP, utilizan este tipo de modelo dentro de sus empresas, y desde ahí realizan las diferentes transacciones hacia los servidores locales o remotos con el sistema SAP [13].

Arquitectura en Nube: La arquitectura en nube utiliza recursos de computación a través de Internet. Los servicios y aplicaciones se alojan en servidores remotos y son accesibles desde cualquier lugar.
Sus principales características son la escalabilidad y flexibilidad, con una gestión centralizada en la nube y pago según su uso.
Ejemplos comunes de este tipo de arquitectura son los servicios de almacenamiento en la nube como Google Drive, servicios de computación como AWS o Microsoft Azure.

Arquitectura de Red Híbrida: Este tipo de red combina elementos de diferentes arquitecturas de red para aprovechar sus ventajas y adaptarse a necesidades específicas.
Las características principales de este tipo de red son la flexibilidad para adaptarse a diferentes requisitos. Puede combinar redes como LAN, WAN, y tecnologías en nube, entre otras. Un ejemplo de esta arquitectura son las redes corporativas que utilizan LAN para conexiones internas, WAN para conectar sucursales y servicios en la nube para aplicaciones y almacenamiento.
Cada tipo de arquitectura tiene sus propias ventajas y desventajas y se elige en función de factores como el tamaño de la red, la distancia geográfica, los requerimientos de velocidad y la escalabilidad.
Los anteriores tipos de red se mencionaron dado que algunos de los clientes, dependiendo del tamaño, rol de sus empresas y número de operaciones de las mismas, podrían tener alguno de ellos instalados dentro de sus instalaciones o entre ellas.

2.6. Servicios

En un sistema operativo, los servicios son los procesos que proporcionan funcionalidades y recursos esenciales para que el sistema y las aplicaciones funcionen de forma correcta. Dichos servicios gestionan el hardware y el software, facilitando permitiendo y realizando diferentes tareas. Los servicios son una especie de intermediarios entre el hardware y los usuarios, facilitando la interacción y el funcionamiento correcto de todo el equipo [15].

Tipos de Servicios en un Sistema Operativo:
Gestión de Procesos:
El servicio de planificación de procesos (scheduler) gestiona la asignación de recursos del CPU además de la ejecución de dichos procesos. Planifica qué procesos deben ejecutarse

y en qué orden. Sincroniza y comunica unos procesos con otros, permitiendo la comunicación entre ellos, a través de semáforos, mutexes y colas de mensajes.

Gestión de Memoria:
El sistema operativo gestiona la memoria del sistema, asignando y liberando espacio en la memoria RAM para los procesos en ejecución. Además, permite a los procesos usar más memoria de la que físicamente está disponible mediante técnicas como la paginación y la segmentación.

Gestión de Archivos:
Proporciona una estructura para almacenar y recuperar archivos en discos u otros medios de almacenamiento manejando la organización, el acceso y la protección de archivos y directorios.

Gestión de entrada/salida (I/O):
Facilita la comunicación entre el sistema operativo y el hardware, como teclados, discos duros o impresoras. El almacenamiento se puede llevar a cabo de forma temporal en el caché, lo que permite mejorar la eficiencia y rendimiento de las operaciones de entrada/salida.

Red:
Implementan los protocolos de red necesarios para la comunicación entre diferentes sistemas a través de redes locales o Internet. Llevan a cabo desde la asignación de direcciones IP hasta gestionar el tráfico de datos.

Seguridad:
Proporcionan servicios de autenticación garantizando que solo los usuarios y procesos autorizados puedan acceder a ciertos recursos. Además, Implementan mecanismos de cifrado para proteger la integridad y privacidad de los datos.

Administración del Sistema:
Recogen y gestionan información sobre el estado del sistema y los eventos que ocurren, facilitando la supervisión y la resolución de problemas. Pueden realizar la automatización de diferentes tareas, permitiendo la programación y ejecución de las mismas.

Windows:
Dentro del sistema operativo Windows, los servicios se llaman tal cual, mientras que dentro del sistema operativo Linux, estos son conocidos como demonios (daemons) que son procesos en segundo plano que realizan tareas como la gestión de redes, impresión, etc.

A pesar de que dependiendo del sistema operativo con el que nos encontremos trabajando, los servicios tienen propósitos similares dependiendo cada uno de ellos, proporcionan el soporte necesario a otros servicios, tareas y además de programas de alto nivel para que funcionen de forma correcta. Algunos de ellos son gestionados diferentemente por el sistema operativo en que se encuentren trabajando, mientras que algunos otros es necesario que sean configurados, programados e inclusive ejecutados de forma manual por el usuario.

2.6.1. Microservicios

Los microservicios son un enfoque arquitectónico para desarrollar aplicaciones de software como un conjunto de servicios independientes y pequeños, en lugar de una única aplicación monolítica. Cada microservicio es un componente autónomo que realiza una función específica y se comunica con otros microservicios a través de interfaces definidas, generalmente mediante APIs (Interfaces de Programación de Aplicaciones). En este enfoque se promueve la modularidad, escalabilidad y flexibilidad en el desarrollo de software [16, 17].

Características:
Modularidad:
Cada microservicio está diseñado para cumplir con una función específica o un conjunto de funcionalidades relacionadas facilitando la comprensión, desarrollo y mantenimiento de cada uno de ellos y la totalidad del sistema del que son parte.

Independencia:
Los microservicios operan de manera independiente y se comunican entre sí mediante protocolos de red (HTTP/HTTPS, gRPC, etc.). Lo que permite que se desplieguen, escalen y se actualicen de forma aislada sin afectar a otros servicios.

Descentralización:
La gestión de datos y la lógica de negocio se distribuyen entre los microservicios en lugar de estar centralizadas en un solo sistema. Cada microservicio puede gestionar su propia base de datos o fuente de datos, lo que permite una mayor flexibilidad y escalabilidad.

Escalabilidad:
Los microservicios pueden ser escalados de manera individual según las necesidades específicas del servicio. Lo que significa que se pueden asignar más recursos a servicios que requieren mayor capacidad sin afectar a otros servicios.

Tecnología Heterogénea:
Cada microservicio puede estar escrito en un lenguaje de programación diferente o utilizar diferentes tecnologías, siempre y cuando permitan la comunicación con otros servicios estos pueden trabajar en conjunto.

Despliegue Independiente:
Los microservicios se pueden desplegar de forma independiente. Permitiendo actualizar y mejorar de forma continua sin necesidad de realizar despliegues masivos que afecten a toda la aplicación.

Resiliencia:
Si un microservicio falla, el impacto en el sistema en general es limitado, ya que otros servicios pueden continuar funcionando, facilitando la construcción de sistemas más resilientes.

Ventajas:
Desarrollo Ágil: Facilita el desarrollo paralelo por equipos pequeños y autónomos. Cada equipo puede trabajar en un microservicio específico sin necesidad de coordinarse estrechamente con otros equipos.

Actualización: Las actualizaciones y mantenimiento se pueden realizar en un servicio específico sin interrumpir el funcionamiento de toda la aplicación.

Escalabilidad: Permite escalar únicamente a los servicios que requieran ser escalados, optimizando el uso de recursos y mejorando la eficiencia.

Flexibilidad: Permite utilizar diferentes tecnologías y lenguajes para diferentes servicios, lo que puede mejorar la eficiencia y adaptabilidad de la aplicación.

Desventajas:
Gestión: La gestión de muchos servicios independientes puede ser compleja, especialmente en términos de coordinación y comunicación entre ellos.

Comunicación en Red: La comunicación entre microservicios a través de la red puede introducir latencia y problemas de rendimiento si no se maneja adecuadamente.

Consistencia de Datos: Mantener la consistencia de datos entre servicios puede ser un desafío, especialmente si cada servicio tiene su propia base de datos.

Seguridad: Cada microservicio puede tener su propio conjunto de riesgos de seguridad que deben ser gestionados de manera independiente.

Los microservicios representan una evolución significativa en la arquitectura de software, ofreciendo flexibilidad y escalabilidad a cambio de una mayor complejidad en la gestión y la integración.

Dentro de la presente propuesta se mencionan los microservicios dado que, se tiene pensado que mediante los microservicios se desarrollen las herramientas para realizar la gestión de cada uno de los servicios pertenecientes al sistema SAP al realizar el encendido o apagado de cada uno de los servidores diferenciando entre cada uno de los sistemas operativos con los que nos encontremos trabajando.

2.7. Seguridad

La seguridad informática en servidores consiste en la aplicación de tecnologías, procesos y buenas prácticas para protegerlos contra accesos no autorizados, fallos de seguridad, pérdida de datos, ataques maliciosos, etc. Esto incluye la protección de los sistemas operativos, servicios, aplicaciones, bases de datos y la infraestructura de red asociada “Server security is about ensuring that the server systems, services, and data are protected from both internal and external threats.” [43,44].

Seguridad Física
La finalidad de la seguridad física es prevenir el acceso físico no autorizado o la destrucción del servidor por causas accidentales o intencionadas llevadas a cabo por alguna persona, a su vez debe considerar los riesgos ambientales o provenientes de la naturaleza. Esto asegura que sólo personal autorizado pueda manipular el hardware o interactuar directamente con el entorno físico del servidor.
Se debe garantizar que algún atacante pueda obtener acceso completo al sistema sin necesidad de vulnerar contraseñas ni firewalls, simplemente conectando físicamente al equipo o extrayendo algún disco duro de los mismos.

Riesgos que mitiga:
- Robo o sabotaje del hardware.
- Manipulación directa del disco duro o memoria.
- Destrucción por fuego, agua, electricidad, etc.

Seguridad del Sistema Operativo
La finalidad de la seguridad del SO es reducir la superficie de ataque del servidor al mantener el sistema operativo actualizado, seguro y configurado con los permisos y servicios mínimos necesarios para funcionar.

Riesgos que mitiga:
- Vulnerabilidades del sistema no parcheadas.
- Escalada de privilegios.
- Accesos a servicios no autorizados o no monitorizados.

El sistema sería fácilmente comprometido por malware, scripts automatizados o atacantes que explotan vulnerabilidades conocidas y documentadas (CVE).

Seguridad de red
La seguridad de la red es una categoría de prácticas y tecnologías que garantizan la protección de las redes internas contra los ataques y las fugas de datos. Comprende el control de acceso, la prevención de los ciberataques, la detección de malware y otras medidas de seguridad.

Riesgos que mitiga:
- Acceso no autorizado.
- Ataques DDoS.
- Robo de información.
- Infecciones de malware.
- Amenazas internas.

Seguridad de aplicaciones
La seguridad de las aplicaciones es un componente crucial de ciberseguridad, que abarca prácticas que evitan el acceso no autorizado, las filtraciones de datos y la manipulación de código del software de aplicación.
Se tiene como objetivo proteger los datos confidenciales y el código de aplicación contra el robo o la manipulación. Esto implica la implementación de medidas de seguridad durante las fases de desarrollo y diseño de la aplicación y el mantenimiento de la protección durante el despliegue y después.

Riegos que mitiga:
- Inyección de código (como inyección SQL o XSS).
- Fugas de información sensible.
- Control de acceso deficiente.
- Autenticación insegura.
- Falta de protección en APIs.
- Exposición a ataques automatizados (bots, scraping).

Seguridad de bases de datos
Las bases de datos son componentes fundamentales en la gestión y almacenamiento de información. Son sistemas organizados que permiten almacenar, gestionar y recuperar datos de manera eficiente. La seguridad de la base de datos se refiere al conjunto de

medidas y políticas diseñadas para proteger la integridad, confidencialidad y disponibilidad de la información almacenada en una base de datos.

Riegos que mitiga:
- Acceso no autorizado a los datos
- Filtración o robo de información
- Modificación o corrupción de datos (integridad)
- Pérdida de información (disponibilidad)
- Inyección SQL

Seguridad lógica o de acceso
La seguridad lógica hace referencia a la seguridad en el uso de los sistemas y del software. Implica también la protección de los datos, los procesos y los programas, así como la del acceso ordenado y autorizado de los usuarios a la información.

Riesgos que mitiga:
- Restringir el acceso a los programas y archivos, sólo a los usuarios autorizados.
- Que los operadores puedan trabajar sin una supervisión minuciosa, pero que no puedan modificar ni programas ni archivos que no correspondan.
- Controlar que se estén utilizando los datos, archivos y programas correctos en y por el procedimiento correcto.
- Garantizar que la información transmitida sea recibida sólo por el destinatario y por ningún otro individuo.
- Asegurar que la información que el destinatario ha recibido, sea la misma que ha sido transmitida.

Seguridad en la nube o entornos virtualizados
La seguridad en la nube es el conjunto de medidas de seguridad cibernética que se usan para proteger las aplicaciones, los datos y la infraestructura basados en la nube. Esto incluye aplicar políticas de seguridad, prácticas, controles y otras tecnologías como la administración de identidades y accesos y las herramientas de prevención de pérdida de datos para proteger los entornos de nube de los accesos no autorizados, los ataques en línea y las amenazas internas.

Riesgos que mitiga:
- Acceso no autorizado a recursos en la nube.
- Filtración o robo de datos.
- Mala configuración de servicios cloud.
- Ataques DDoS sobre servicios virtualizados o públicos.
- Vulnerabilidades en APIs expuestas
- Aislamiento deficiente entre entornos virtuales o inquilinos

Copia de seguridad y recuperación
La copia de seguridad y la recuperación son prácticas de gestión de datos diseñadas para proteger los datos frente a la pérdida o la corrupción y garantizar su disponibilidad en caso de desastre.
La forma de hacer copias de seguridad de los datos y aplicaciones depende del costo de perder el acceso a los datos y aplicaciones durante cualquier período de tiempo y del costo de reemplazarlos si se pierden definitivamente.

Riesgos que mitiga:
- Pérdida de información por fallos técnicos.

- Ataques de ransomware.
- Errores humanos (borrado accidental).
- Sabotaje interno o externo.
- Desastres naturales o siniestros físicos.
- Interrupción prolongada de servicios.

2.7.1. JWT

JWT (JSON Web Token) es un método para representar claims (declaraciones) de forma segura entre dos partes. Es parte del conjunto de especificaciones de seguridad de OAuth 2.0 y OpenID Connect, y su principal uso es en autenticación y autorización.
Un JWT permite que un servidor genere un token, que luego el cliente (navegador, app, etc.) puede usar para hacer peticiones autenticadas, sin necesidad de enviar usuario y contraseña en cada solicitud. El monitoreo y registro detecta, analiza y responde a incidentes de seguridad en tiempo real, además de generar trazabilidad para auditorías y análisis forense.

Riesgos que mitiga:
- Actividades sospechosas sin detección.
- Ataques prolongados (APT) que pasan desapercibidos.
- Imposibilidad de investigar un incidente tras su ocurrencia.

Tipos:
Los JSON Web Tokens (JWT) son tokens de seguridad que se pueden firmar o cifrar. Los principales tipos de JWT son la Firma Web JSON (JWS) y el Cifrado Web JSON (JWE).

- Firma web JSON (JWS): el contenido de este tipo de JWT se firma digitalmente para asegurarse de que el contenido de la JWT no se altere en tránsito entre el remitente y el destinatario. El contenido o las reclamaciones del JWS pueden ser legibles también por otras partes. Por lo tanto, se puede utilizar un JWS para verificar la integridad del contenido o reclamación, pero no se debe utilizar para transferir datos confidenciales como contraseñas. JWS se utiliza normalmente a través de HTTPS o SSL porque no impide intrínsecamente la lectura de los datos.

- JSON Web Encryption (JWE) : el contenido de este tipo de JWT se cifra digitalmente. Esto significa que se puede utilizar para verificar la integridad y proteger el contenido. Puede utilizarse sobre HTTP plano, ya que encripta intrínsecamente el contenido.

Algoritmos utilizados:
Los algoritmos definen cómo se firma el JWT. Se dividen en simétricos y asimétricos.
Algoritmos Simétricos (misma clave para firmar y verificar):
- HS256 (HMAC con SHA-256)
- HS384
- HS512

Se utiliza una clave secreta compartida. Es más rápido, pero menos seguro si la clave es expuesta.

Algoritmos Asimétricos (clave privada para firmar, pública para verificar):
- RS256, RS384, RS512 – Basado en RSA

- ES256, ES384 – Basado en curvas elípticas (ECDSA)
- PS256 – RSA con padding probabilístico (más seguro)

Ideal para escenarios donde diferentes servicios validan el token sin conocer la clave privada.

Características principales:

- Compacto: Ideal para usar en URLs, cabeceras HTTP o almacenamiento local.
- Seguro: Firma digital garantiza la integridad (pero no cifra el contenido, salvo que se use JWE).
- Stateless: El servidor no guarda estado entre peticiones (sin sesión en memoria).
- Escalable: Útil para sistemas distribuidos y microservicios.
- Seguridad basada en firma: Garantiza integridad (no confidencialidad a menos que se use cifrado con JWE).
- Autocontenible: Toda la información necesaria está dentro del token. No se consulta la base de datos.

3. Estado del Arte

3.1. Client/Server Remote Control Administration System: Design and Implementation

Las redes de computadoras están en todas partes implementadas y en todos los sectores, incluido el de TI, industrial, de gestión y es necesario acceder a ellos de forma remota. La tecnología centraliza y organiza para mantener los costos bajos, muchos sitios remotos se quedan sin soporte de TI en el sitio. Computadora remota. Ventajas; Muchas actividades se pueden completar y no es necesario realizarlas accedido personalmente por el administrador. El sistema de administración de control ha sido diseñado e implementado en base a la arquitectura cliente/servidor. El objetivo principal del sistema propuesto permite a los clientes solicitar ayuda desde servidores remotos a través de la red [5]. En la figura 6, podemos ver el diagrama descriptivo del sistema.

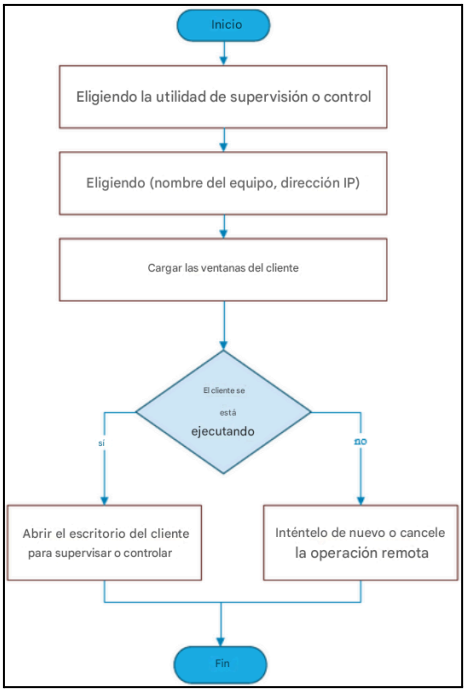


Fig 6. Diagrama de flujo del sistema. Tomada de [5].

3.2. Design and Implementation of an Administration System for Distributed Web Server

El crecimiento explosivo de la World Wide Web ha suscitado gran preocupación respecto de muchos desafíos: rendimiento, escalabilidad y disponibilidad del sistema web. En consecuencia, Web, los creadores de sitios construyen cada vez más sus servidores web como sistemas distribuidos para resolver estos problemas, y es probable que esta tendencia se acelere. En tales sistemas, un grupo de los hosts acoplados trabajarán juntos para servir como un único servidor virtual. El servidor puede proporcionar un rendimiento atractivo y adaptarse al crecimiento del tráfico web, inevitablemente aumenta la complejidad de la administración del sistema. En este artículo exploramos las ventajas de Java para diseñar e implementar un sistema de administración para abordar este problema [6]. En la figuras 7 y 8, podemos ver el diagrama descriptivo del sistema.

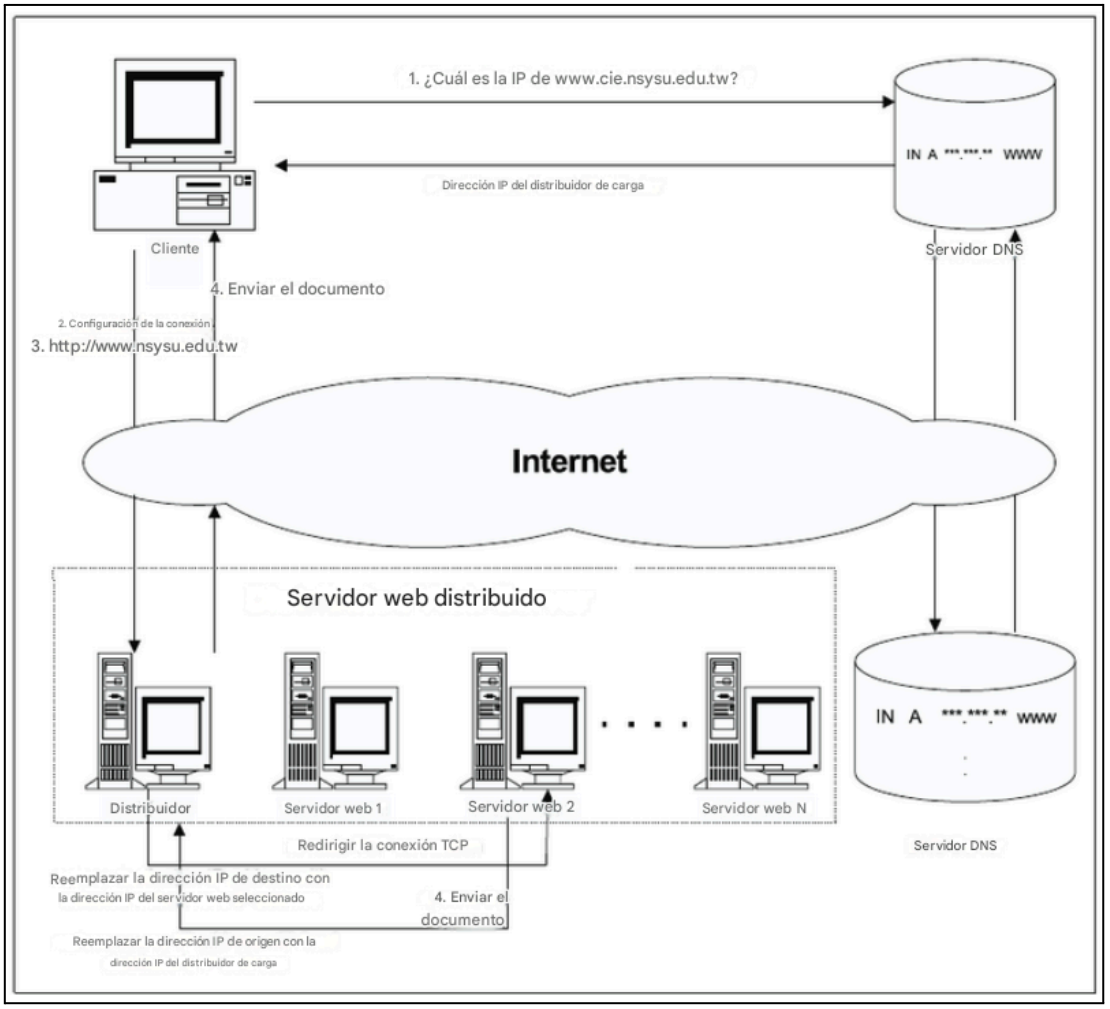


Fig 7. Descripción general del servidor web distribuido. Tomada de [6].

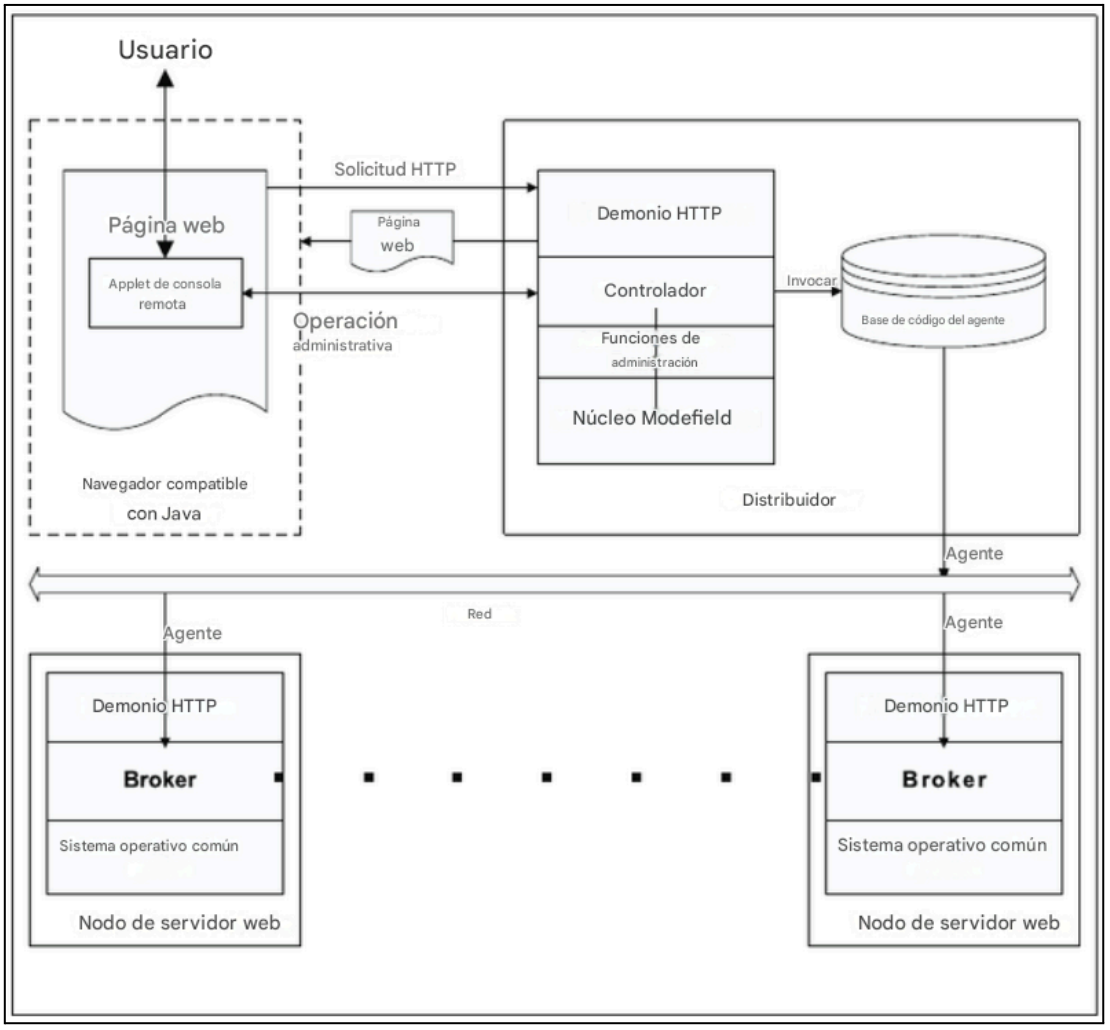


Fig 8. Arquitectura general del sistema propuesto. Tomada de [6].

3.3. Optimización de costos para servicios SAP en AWS con plataforma NovisCloud

NovisCloud juega un rol esencial para mejorar la disponibilidad de SAP en AWS a través de la integración de métricas de monitoreo de SAP en CloudWatch (plataforma de monitoreo de AWS) y la implementación de mecanismos automatizados de recuperación de los servicios ante distintos escenarios de falla (auto-healing), extendiendo las capacidades de recuperación nativas de AWS hacia las aplicaciones SAP. Por medio de esta plataforma, es posible realizar orquestación de procesos de recuperación y validaciones automatizadas de los sistemas recuperados.

Los sistemas basados en el servidor de aplicación SAP Netweaver ABAP tienen una arquitectura monolítica o stateful, esto implica que la información de la sesión o estado de cada proceso, son almacenados en el servidor de aplicación donde es procesado. Cuando un usuario se conecta a SAP, mediante SAPGUI o navegador, la aplicación realiza un proceso de balanceo y direcciona el usuario a un servidor de aplicación, donde permanece conectado, hasta que finaliza su sesión (algunos usuarios suelen estar todo el día trabajando en la misma sesión, conectado al mismo servidor de aplicación). Esta

característica provoca que la caída de un servidor de aplicación impacte directamente el trabajo de un usuario o proceso. El usuario (o proceso) afectado tendrá que volver a iniciar sesión (o relanzar el proceso) y reanudar el trabajo desde la última transacción no finalizada. En la siguiente imagen, la figura 9, podemos ver el diagrama descriptivo del sistema.

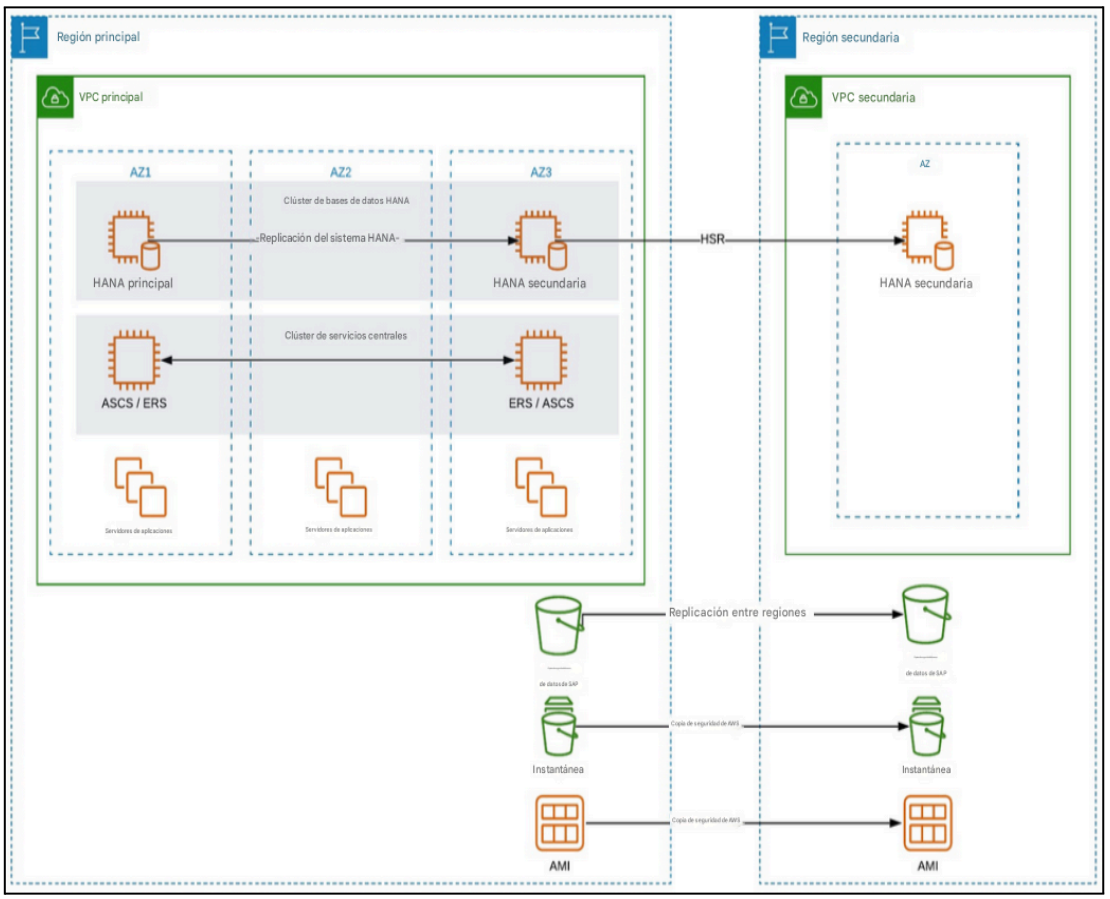


Fig 9. Arquitectura avanzada. Tomada de [9].

Novis propone dos arquitecturas alternativas que a pesar de no igualar los SLAs de la arquitectura anterior, entregan niveles de servicio elevados comparados con soluciones On Premise y permiten obtener importantes reducciones en costos. La arquitectura 2 involucra la replicación de la base de datos a una instancia de menor tamaño y la replicación de los servidores de aplicación utilizando CloudEndure. La arquitectura 3 utiliza las capacidades nativas de recuperación de AWS para restablecer el servicio ante una falla de una instancia y la redundancia de los almacenamientos para respaldos (AWS S3 y EBS snapshots) para la recuperación de los servicios ante la falla de una Availability Zone [9].

3.4. OpManager

Es un software confiable de gestión de redes y servidores, monitorea activamente más de 300 métricas que son críticas para el rendimiento del servidor, además de los servicios y procesos, ofrece a los administradores de TI una única consola de gestión que brinda una total visibilidad y control de los servidores físicos y virtuales [7]. En la figura 10, podemos ver el diagrama descriptivo del sistema.

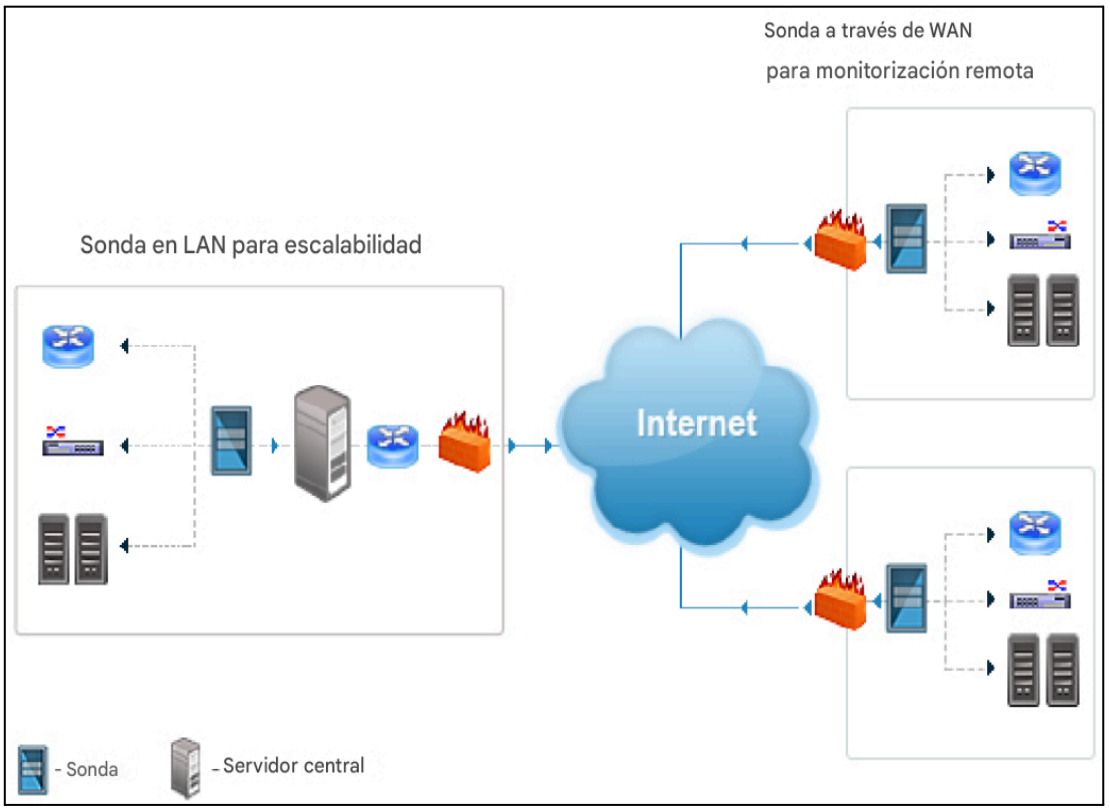


Fig 10. Arquitectura de OPManager Edición Enterprise. Tomada de [7].

Este nos permite:

- Programar las actividades de mantenimiento en sus servidores de forma periódica.
- Permite acceder a los servidores de forma remota dentro de su cliente web. Puede conectarse al servidor directamente utilizando las credenciales SNMP, WMI, Telnet / SSH con la conexión de Remote Desktop o conectarse a un servicio web en particular que se ejecuta en el servidor a través de una conexión HTTP o HTTPS.
- Automatización de la gestión de servidores por medio de flujos de trabajo, por ejemplo:
    - Iniciar / detener el mantenimiento de un servidor.
    - Iniciar / Detener / Pausar / Reiniciar los servicios de Windows.
    - Detener / Apagar / Reiniciar el SO invitado.
    - Actualizar el almacén de datos en los servidores virtuales.

4. Planeación

4.1. Antecedentes del Proyecto

La presente propuesta de tesis surgió como solución a un problema que el consultor de TI-Senior, Lic. Antonio Martínez Alvarado tiene con respecto a las arquitecturas que sus clientes mantienen en renta dentro de la nube de AWS que utilizan el sistema SAP SE. En el campo laboral en el que se desempeña (proveedor de servicios SAP) él ofrece a sus

clientes la posibilidad de aminorar los costos de administración de sus arquitecturas en la nube al ofrecer este servicio, permitiéndoles ahorrar en este sentido además de en los costos asociados por la renta de dichos servidores, al encargarse del encendido y apagado manual de cada servidor según las indicaciones de sus clientes. De manera que cada cliente decide en qué tiempos necesita que sus servidores permanezcan encendidos y cuando le conviene que se apaguen, lo que muy frecuentemente sucede por las noches y fines de semana dependiendo del giro del negocio de los clientes, de manera que se logre maximizar el ahorro de los clientes minimizando costos en los precios de la renta y salarios asociados a un departamento de TI dentro de las empresas de los clientes, según sea el caso. El Lic. Antonio, se desempeña como proveedor de servicios SAP a diferentes niveles, desde la parte de asesoramiento en los servicios SAP, hasta la parte de configuración y migración de infraestructuras físicas SAP SE locales a servidores en la Nube.
En el proceso de entrevista para el levantamiento de requerimientos, se detectó una deficiencia en los servidores de producción en la nube utilizados por la plataforma SAP SE, en los cuales se tiene montada dicha infraestructura. Para soportar la cantidad de procesos realizados dentro de dichos sistemas, la cantidad de servidores de producción utilizados es directamente proporcional a la cantidad de trabajo que se tenga que procesar en los mismos. Esto implica que en la mayoría de los casos los horarios de mayor carga de trabajo en estos servidores corresponden a un horario de oficina normal de 9:00 am a 6:00 pm. por lo que no tiene sentido que los servidores contratados en la nube continúen operando si la carga de trabajo es nula o demasiado baja dependiendo del uso que cada uno de los clientes les da. Es por eso que se propone como solución a este problema un sistema de administración remota de servidores de producción en la nube de AWS para minimizar en lo posible los costos operativos que invierten los clientes en la infraestructura descrita. Cabe mencionar que ésta propuesta de solución también considera la administración del ciclo de vida (encendido, apagado) de los servidores de los hipotéticos clientes de forma diferenciada e independiente, siempre tomando en cuenta la forma en la que los procesos de trabajo se llevan a cabo puesto que es de vital importancia el orden en cómo se ejecuten cada uno de ellos y así evitar la corrupción o pérdida de los datos.

4.2. Análisis del problema

En esta etapa se comienza con la investigación de los tipos de negocios o clientes finales a los que la presente tesis puede ayudar o en donde se puede aplicar. Después se realiza una descripción de los perfiles de dichos clientes tomando en cuenta sus características por en qué base a ello se diseñó la presente solución.

4.2.1. Perfiles de casos de estudio

En esta sección se realizó el análisis de los perfiles de negocios a los que va dirigido este trabajo de tesis. Se comenzó realizando los tipos de ambientes dentro del entorno SAP, después se continuó con el análisis de ..

- **Perfil A: Ambientes No Productivos (DEV/QAS/Sandbox)**

En el contexto de la gestión de sistemas SAP, los ambientes no productivos DEV (Desarrollo), QAS (Garantía de Calidad) y Sandbox se definieron como componentes esenciales de una arquitectura de tres niveles, diseñada para gestionar los cambios de software de manera controlada y mitigar riesgos antes de que afectarán las operaciones empresariales en el ambiente Productivo (PROD). Estos son sistemas separados que se utilizan para diferentes propósitos antes de que los cambios lleguen al ambiente Productivo (PROD), donde se realizan las operaciones comerciales reales.

Desarrollo (DEV): Su propósito es la creación y modificación de código ABAP, personalización funcional y configuración del sistema. En este ambiente, se puede detener el trabajo de los desarrolladores, pero no impacta las operaciones de negocio. Este ambiente es ideal para apagado programado nocturno y fines de semana.

Calidad/Pruebas (QAS): Dentro de sus propósitos se encuentran las pruebas unitarias, pruebas de integración, pruebas de regresión y capacitación de usuarios clave. Este ambiente puede ser el motivo de retrasar el ciclo de liberación (release cycle) de cambios, impactando indirectamente al negocio. Este ambiente puede ser programado para el apagado, con posibles excepciones para pruebas extendidas.

Producción (PRD): Su propósito es dar soporte a las operaciones diarias del negocio (ventas, finanzas, logística, manufactura). En este el tiempo de inactividad genera pérdidas directas de negocio (ingresos, multas, interrupción de procesos). Este ambiente, está limitado a ventanas de mantenimiento muy cortas o procesos de recuperación/contingencia.

Sandbox: Este es un ambiente de libre acceso, volátil y sin necesidad de backup regular. Su patrón de uso es intermitente y bajo demanda. El ahorro en este ambiente es máximo, ya que pueden estar inactivos el 90% del tiempo.

Training (TRN): Utilizado sólo para capacitación. Se mantiene inactivo y se enciende una o dos veces al mes. El ahorro en este ambiente es máximo. La política de Start/Stop debe ser activada por una programación manual o un evento de calendario.

Pre-Production (PPRD): Un staging final antes de PRD. Suele estar activo sólo durante el ciclo de liberación. Su patrón de uso suele ser semanal o mensual y el ahorro en este ambiente es alto, ya que puede apagarse entre ciclos de release.

Los sectores principales en los que podemos encontrar este tipo de ambientes son: Manufactura, Retail, Servicios Financieros, etc. Dentro de estos sectores, dichos entornos se utilizan para el desarrollo de customizing, pruebas de regresión, integración y patches.
Su uso está intrínsecamente ligado al horario laboral del equipo técnico o funcional. Esto crea una ventana de inactividad de 14-16 horas diarias durante la semana, más las 48-60 horas del fin de semana. La disponibilidad requerida (SLA) para estos ambientes se contempla 15-20%, contrastando fuertemente con la disponibilidad de la infraestructura EC2 de AWS que se factura $24/7 por lo que si el presente trabajo se aplica en este sector, el ahorro del cliente final sería significativo.

- **Perfil B: Clientes de Operación Diurna/Monoturno**

Este perfil tiene un alto grado de optimización de costos en infraestructura productiva, ofreciendo un equilibrio entre la necesidad de disponibilidad y la oportunidad de ahorro. En este perfil podemos encontrar empresas de Manufactura, Distribución, Logística Local o Comercio Minorista (Retail) que operan principalmente dentro de una jornada laboral bien definida, típicamente de 10 a 16 horas al día. La principal característica de este perfil es la existencia de una ventana de inactividad de negocio durante la noche. Mientras que las operaciones transaccionales (ventas, entradas de mercancía, producción) cesan, la infraestructura SAP en AWS continúa encendida y facturando.

- **Perfil C: Clientes con Ambientes de Capacitación/PoC**

Este perfil se centra en la eficiencia, el control de costos y la gobernanza (FinOps). Este perfil engloba a empresas de Consultoría, *Holding Groups o grandes corporativos que utilizan AWS para proyectos temporales o de demostración. Con este tipo de proyectos, el uso de estos ambientes es intermitente y esporádico, completamente desligado del ciclo de producción diario. Suele utilizar estos ambientes para Proof of Concept (PoC) los cuales son ambientes que se levantan para probar una nueva funcionalidad S/4HANA o una integración, se usa durante 4 a 8 horas únicamente. También suelen utilizarse para impartir cursos de una semana, pero permanece encendido las 3 semanas restantes del mes por si es necesario ampliar el desarrollo de alguna prueba extra.

Casos de estudio seleccionados:

Dado el análisis realizado de los perfiles de cada uno de los ambientes dentro del entorno SAP, se seleccionaron los siguientes casos de estudio.

- **Caso 1: Ferretería (Cadena de Tiendas Local)**

Este caso de estudio fue seleccionado dado que entra dentro de la clasificación del perfil b (Operación Diurna/Monoturno) y está enfocado en el Sistema Productivo.
Las cualidades que los definen son:
Horario Fijo: Las tiendas tienen un horario de atención claro (ej. 8:00 AM a 6:00 PM). Las transacciones SAP (ventas, inventario) se concentran en estas horas.
Procesos *Batch Críticos: Se llevan a cabo una vez que el horario de atención se encuentra concluido, estos contempla el cierre de caja, conciliación de inventarios y la actualización de precios desde la central.

- **Caso 2: Cadena de Tiendas de Conveniencia (24/7)**

Este caso de estudio fue seleccionado dado que entra dentro de la clasificación del perfil a (Ambientes No Productivos DEV/QAS).
PRD 24/7: Este caso se centra en el sistema Productivo, no puede apagarse bajo ninguna circunstancia dado que tiene una operación continua. Para este caso el foco de ahorro se traslada a los ambientes de desarrollo y calidad dado que este tipo de negocios se caracteriza por integrar constantes promociones, cambios de inventario y adaptaciones regulatorias, el equipo de TI tiene una intensa actividad de desarrollo y pruebas.
Dicho lo anterior, para este caso, la métrica de ahorro es directamente proporcional a los desarrolladores que trabajan en dichos negocios, éstos usualmente tienen un horario laboral

de Lunes a Viernes (9:00 AM - 6:00 PM), por lo que fuera de estos horarios y en fines de semana las instancias pertenecientes a los ambientes de DEV y QAS, deberían permanecer apagadas logrando un ahorro predecible.

- **Caso 3: Cadena de Tiendas de Comida Orgánica en Crecimiento**

Este caso de estudio fue seleccionado dado que entra en la clasificación del perfil c (Capacitación y Ambientes de Proyectos Especiales). Este tipo de negocios, permanecen siempre en crecimiento, hay alta demanda de capacitación de personal nuevo y constantes proyectos de expansión/PoC por ejemplo la integración de nuevos *marketplaces o sistemas de *delivery*. Es por esto que se deben crear ambientes SAP temporales con frecuencia para estos proyectos.

4.3. Estrategia de solución

La presente tesis propone una mejora en la administración de servidores alojados en la nube de AWS para SAP SE. AWS (Amazon Web Services) es una colección de servicios de computación en la nube pública, ofrecidas a través de Internet por la empresa Amazon.com.
Por otra parte, SAP es la sigla del nombre alemán original de la empresa: Systemanalyse Programmentwicklung, que se traduce como "desarrollo de programas de sistemas de análisis". Hoy en día, el nombre corporativo legal de la empresa es SAP SE, donde SE significa "Societas Europaea", que significa que es una empresa pública registrada de conformidad con la legislación corporativa de la Unión Europea. SAP SE es un ERP ("Enterprise Resource Planning") que se traduce como "planificación de recursos empresariales".
El software de ERP incluye programas para todas las áreas de negocio centrales, tales como compras, producción, gestión de materiales, ventas, marketing, finanzas y recursos humanos (RR. HH.).
Se propone el desarrollo de una plataforma web alojada en la nube de AWS, que nos permitirá administrar la infraestructura (servidores y servicios utilizados para SAP SE) de una lista de clientes hipotéticos. Dichos clientes podrían tener dentro de sus arquitecturas en la nube diferente cantidad de servidores además de diferentes gestores de bases de datos.
El objetivo es tratar de mantener al mínimo el tiempo de renta de cada uno de los servidores de los hipotéticos clientes sin afectar las cargas de trabajo que a cada uno de ellos convenga, es decir, se supone que un cliente tiene en renta dentro de la nube de AWS un servidor que utiliza junto a SAP SE para manejar su producción diaria dentro de una de sus sucursales. Ahora suponemos que es una sucursal pequeña, por lo cual únicamente utiliza y le es suficiente con un solo turno laboral de 8 horas de producción, por lo cual no es necesario que el servidor asociado en particular a esta sucursal del cliente permanezca encendido las 24 horas del día considerando que solo lo utiliza durante 8 horas cada día. Considerando este panorama, se propone mediante el presente trabajo administrar el encendido y apagado de manera segura del servidor o servidores además de los servicios asociados a la infraestructura de SAP SE del cliente de acuerdo con sus conveniencias siempre teniendo en cuenta el ahorro de dinero por la renta de los servidores en la nube de AWS. Se plantea que la reducción no solo se de a nivel monetario si no operacional al administrar de mejor manera las infraestructuras de los clientes.
Los clientes que utilizan el ERP SAP SE pueden gestionar mediante el mismo diferentes áreas de sus empresas como lo son: compras, producción, gestión de materiales, ventas, marketing, finanzas y recursos humanos (RR. HH.) es por esto que las industrias que han

integrado este software a sus empresas son diversas entre las cuales destacan: aeroespacio y defensa, agronegocios, automotriz, banca, productos químicos, productos de consumo, construcción, bienes raíces, defensa y seguridad, gobierno, alta tecnología, educación superior e investigación, fabricación industrial, seguros, ciencias de la vida, cuidado de la salud, medios, deportes y entretenimiento, productos manufacturados, minería, petróleo, gas y energía, servicios profesionales, comercio minorista, telecomunicaciones, viajes y transporte, servicios públicos, distribución mayorista, etc.
Es por esto que el campo de aplicación de la presente propuesta es amplio y basto, lo que representa diferentes oportunidades a la vez que grandes retos, es decir dado los diferentes campos de aplicación y diferentes industrias, la información que los servidores contienen o manejan en su día a día es de sensible a muy sensible por lo cual las conexiones que se manejen dentro de la presente propuesta deben estar cifradas y se debe considerar que los servidores y datos con los que trabajaremos serán una simulación que represente a los servidores y datos reales, dado que por el momento este trabajo aún no se puede implementar para la administración de la infraestructura de un cliente real por lo ya mencionado.

4.3.1. Análisis y Modelado Formal de la solución

Análisis de Requisitos Funcionales y No Funcionales (RNF)

- **Servidor Bastión Orquestador (SBO)**
  El servidor debe satisfacer requisitos estrictos para garantizar la integridad operativa de los ambientes SAP y la seguridad del acceso. El RNF crítico de Rendimiento se define por el Tiempo Total de Ejecución (TPE), el cual debe ser <30 segundos para el 95% de los ciclos de vida (Start/Stop). El RNF de Seguridad se fundamenta en la implementación del Control de Acceso Basado en Roles (RBAC) de tres niveles (0, 1 y 2), asegurando que solo los usuarios Nivel 1 puedan operar los servidores asignados, y solo el Nivel 2 pueda gestionar usuarios y entidades. El RNF de Confiabilidad se mide por la Tasa de Fallos en el Ciclo de Vida (TFC), cuyo umbral de aceptación es <1%.

Modelado de la Arquitectura y Despliegue

- **Descripción Inicial**
  La arquitectura del SBO se formaliza como un nodo único de alto aislamiento, utilizando el Diagrama de Despliegue UML para representar el Windows Server 2019 Datacenter y su SQL Server dedicado. El SBO reside en el dominio orquestador.top, sirviendo como el único punto de acceso para la gestión y la visualización. La modularidad interna del software (Python) se especifica a través del Diagrama de Componentes UML , detallando las interfaces (I_Acceso_Credenciales) requeridas entre el Módulo Core, MonitorPy y el Repositorio SQL.

Modelado de Datos y Control de Acceso

- **Descripción Inicial:**
  El diseño de la persistencia de datos se documenta en el Diagrama de Clases UML, centrado en las entidades Usuario, Negocio, InstanciaEC2 y Credenciales. La entidad InstanciaEC2 incluye el atributo orden_secuencia y estado_actual, esenciales para la secuencia crítica (Start/Stop) y la actualización por MonitorPy, respectivamente. Se garantiza que la estructura de la base de datos SQL soporta la recuperación de metadata con una latencia <100 ms para cumplir con los requisitos de rendimiento y asegurar que la clave del servidor asociado a cada cliente se encuentre disponible en forma segura.

4.3.2. Desarrollo de la solución

En esta sección se documentó la construcción del software y la configuración del entorno del sistema.

- **Configuración de la Plataforma de Ejecución:**
  La plataforma se implementa sobre una instancia de Windows Server 2019 Datacenter en AWS, utilizando una base de datos tipo SQL para el almacenamiento de la metadata. El SBO aloja el intérprete de Python 3.x y las dependencias necesarias. La configuración de red incluye la asignación del dominio orquestador.top al API REST del servidor, garantizando que todo acceso de usuario y operación remota utilice este endpoint central y esté protegido por HTTPS/TLS.

- **Desarrollo del Módulo de Seguridad y RBAC:**
  El Módulo de Seguridad implementa la autenticación basada en JSON Web Tokens (JWT) para la interfaz web. El RBAC se integra a nivel de la API REST, donde cada solicitud de operación verifica el token y el nivel de acceso al que el usuario en cuestión tiene acceso, esto lo revisa contra la base de datos. Se valida que el sistema deniega el 100% de los comandos a usuarios Nivel 0 (visualización) y permite el 100% de la funcionalidad a los usuarios Nivel 1 para sus servidores designados.

- **Desarrollo del Módulo Core y Trazabilidad:**
  El Módulo Core (Python) implementa la lógica de la secuencia crítica del Diagrama de Secuencia UML. Utiliza las Credenciales recuperadas del SQL Server para ejecutar comandos remotos en los servidores de producción de los clientes. El Módulo de Trazabilidad registra cada evento de Start/Stop en el Repositorio SQL, incluyendo el usuario y el tiempo total, logrando una tasa de registro del 100%.

- **Desarrollo del Programa MonitorPy:**
  El programa MonitorPy (Python) se implementa como un servicio continuo, según el Diagrama de Actividades UML. Su función es reemplazar a cualquier servicio de monitoreo externo. El script corre en un ciclo infinito de forma continua, consulta la

DB por las credenciales, se conecta remotamente a los servidores de producción, obtiene el estatus real, y actualiza el campo estado_actual en la tabla InstanciaEC2 solo si detecta un cambio. Esto garantiza que el estado mostrado en la interfaz web tiene una latencia de actualización <5 segundos.

4.4. Metodología

A continuación se describe la metodología seguida para la realización de la presente tesis, esta fue desarrollada en cuatro fases formales de Ingeniería de Software, enfocándose en el modelado, la implementación funcional, la validación de operación exitosa así como la cuantificación económica.

Fase 1: Análisis, Modelado Formal y Especificación.

- **Análisis Profundo del Problema Operativo.**
  - Identificación de subutilización: Se cuantificó de forma precisa las horas en que los ambientes SAP no productivos (Desarrollo, Pruebas, Calidad) permanecen activos sin uso, sirviendo como base para los Casos de Estudio.

- **Análisis de Requisitos Funcionales y No Funcionales:**
  - Funcionales: Definición precisa de los comandos Start/Stop y la lógica de la secuencia crítica (SO → DB → Application).

  - No Funcionales: Establecimiento de los RNF clave: Seguridad (RBAC Nivel 0, 1, 2), Disponibilidad (99.9% del SBO), Rendimiento (TPE < 30 segundos) y Trazabilidad.

- **Modelado de la Arquitectura y Despliegue:**
  - Se formaliza la arquitectura del sistema mediante el Diagrama de Despliegue UML, especificando los nodos físicos (Windows Server 2019/SQL Server) y la topología de red (Dominio orquestador.top).

  - Se utiliza el Diagrama de Componentes UML para definir la modularidad del software (Core, MonitorPy, Security) y sus interfaces requeridas y provistas (I_Acceso_Credenciales), asegurando una estructura limpia y mantenible.

- **Modelado de Datos y Control de Acceso:**
  - Se diseña el Diagrama de Clases UML para establecer la estructura de la Base de Datos SQL Server. Este diagrama es vital para modelar la jerarquía Usuario (Nivel 0, 1, 2) → Negocio → AmbienteSAP → InstanciaEC2.

  - Se especifican las clases Usuario y Credenciales para cumplir con la separación de responsabilidades y el acceso seguro a los datos por parte de MonitorPy y el Módulo Core.

Fase 2: Implementación, Desarrollo y Prueba de Interfaces.

- **Configuración de la Plataforma de Ejecución:**

- Instalación y hardening del Windows Server 2019 Datacenter para alojar el SBO y el SQL Server dedicado.

- Configuración del entorno Python y el ruteo del dominio orquestador.top al API REST del SBO.

- **Desarrollo del Módulo de Seguridad y RBAC:**
  - Implementación del esquema de autenticación JWT y 2FA y las rutinas de gestión de usuarios (exclusivas del Nivel 2).

  - Desarrollo de la lógica de RBAC para que la API REST consulte los derechos del usuario (nivel_acceso) y garantice la Cobertura de Funcionalidad del 100% de los comandos solo para usuarios Nivel 1.

- **Desarrollo del Módulo Core y Trazabilidad:**
  - Codificación de las rutinas Python que implementan la lógica del Diagrama de Secuencia UML para los comandos Start/Stop a través de la conexión remota (SSH/WinRM).

  - Implementación del Módulo de Trazabilidad (Logger) que garantiza el registro del 100% de los eventos (quién, qué, cuándo, resultado) para el cumplimiento de auditoría.

- **Desarrollo del Programa MonitorPy:**
  - Implementación del Diagrama de Actividades UML para crear el script Python de monitoreo continuo.

  - El script debe consultar la DB SQL para obtener las credenciales y el estado anterior, conectarse al servidor de producción, y actualizar el estado en la DB solo si se detecta un cambio, asegurando la Latencia de Actualización <5 segundos para la interfaz.

Fase 3: Pruebas de Rendimiento y Confiabilidad.

- **Pruebas de Rendimiento:**
  - Se ejecutan pruebas de estrés asumiendo una operación exitosa. Se mide el Tiempo Total de Ejecución (TPE).

  - Criterio de Éxito: Se valida que el TPE es < 30 segundos en el 95% de las pruebas.

- **Pruebas de Confiabilidad Operacional:**
  - Se realizan ciclos repetitivos y controlados para medir la Tasa de Fallos en el Ciclo de Vida (TFC). Los fallos aquí se limitan a errores de sintaxis, conexión

inicial o permisos (errores del propio SBO), no a errores de lógica de *rollback*.

- Criterio de Éxito: Se valida que la TFC es < 1%, demostrando la ejecución exitosa de las secuencias.

Fase 3: Cuantificación del Impacto y Conclusiones.

- **Cálculo Financiero:**
  - Se documenta el Costo Base Operativo Semanal (CB) (24/7) para los Casos de Estudio.
  - Se calcula el Costo Optimizado Semanal (CO), aplicando los horarios de operación definidos por el SBO.
  - Validación de Hipótesis: El cálculo debe demostrar un ahorro de costos > 35% en el escenario de mayor subutilización.
- **Correlación Final y Conclusión:**
  - Se presenta la matriz de resultados correlacionando las métricas de QoS (TFC) con el Ahorro Económico (%).
  - Conclusión Formal: La tesis concluye que el SBO logra una optimización de costos >35% sin comprometer la confiabilidad operativa (TFC <1%) en las operaciones exitosas.

4.5. Aportaciones

Las aportaciones que el presente proyecto realiza a las ciencias, son:

- El modelado, diseño e implementación del simulador de orquestación de servidores y servicios en la nube.
- El control remoto de cada uno de los servidores de producción en la nube de AWS del tipo Windows Server con los que cada uno de los clientes cuenta dentro de sus infraestructuras de tamaño dinámico.
- El control en el apagado de cada uno de los servicios montados en los servidores de producción en un orden en específico, dependiendo de la versión del sistema SAP con la cuenten los clientes dentro de sus empresas.
- El control en el encendido de cada uno de los servidores que se requiera habilitar, tomando en cuenta el orden de encendido de los servicios con los que cuenta cada servidor de producción.

4.6. Infraestructura

4.6.1. Hardware

- **1 Laptop Intel(R) Core(TM) i7-8560U CPU @ 1.80GHz**
  Equipo de cómputo personal, el cual será utilizado en el desarrollo de cada una de las etapas del proyecto. En dicha computadora se realizará la captura de requerimientos, diseño de los diagramas de UML, diseño de las pruebas unitarias y el desarrollo de software.
  A través de esta pc, se realizará la configuración de los servidores bastión y producción, mismos que representarán a los servidores reales utilizados por cada uno de los clientes para el manejo de sus cargas de trabajo.
  A su vez se utilizará para realizar la conexión remota hacia los servidores previamente configurados en el esquema antes definido.

- **Servidor Bastión instancia EC2 AWS con Microsoft Windows Server**
  Servidor Virtual Windows montado en la nube de AWS, el cual contendrá una base de datos con todos los datos de los clientes hipotéticos, los cuales para el presente trabajo se limitarán a dos clientes. Cada uno de los cuales trabajará con un sistema operativo diferente, además de diferentes motores de bases de datos. Esto con el fin de representar de la mejor manera posible lo heterogéneo de las arquitecturas de cada cliente aplicando un método inductivo resolviendo los retos de una arquitectura particular y de ahí dirigirse hacia lo general.
  Este servidor permanecerá prendido en un formato 24/7, para siempre proveer conexión hacia la plataforma web de los clientes, además de monitorear a los servidores de producción que se mencionan a continuación.

- **Servidor de producción instancia EC2 AWS con Windows Server 2019**
  Servidor con el sistema operativo Windows que recibirá las cargas de trabajo generadas por un programa que genera datos aleatorios, los cuales se irán almacenando dentro de la base de datos contenida dentro del mismo. Dicho servidor manejará todos los datos concernientes al negocio de un cliente.
- **Servidor de producción instancia EC2 AWS con Linux Ubuntu**
  Servidor con el sistema operativo Linux Ubuntu que recibirá las cargas de trabajo generadas por un programa que genera datos aleatorios, los cuales se irán almacenando dentro de la base de datos contenida dentro del mismo. Este servidor manejará todos los datos del negocio de un cliente.
- **Servidor de producción instancia EC2 AWS con Windows Server 2022**
  Servidor con el sistema operativo Windows Server que recibirá las cargas de trabajo generadas por un programa que genera datos aleatorios, los cuales se irán almacenando dentro de la base de datos contenida dentro del mismo. Este servidor manejará los datos del negocio de otro cliente registrado en la plataforma.

4.6.2. Software

- **CMD** Windows
  El CMD o Command Prompt, es una interfaz de línea de comandos en sistemas operativos Windows, esta nos permite interactuar con el sistema operativo mediante comandos de texto en lugar de usar la interfaz gráfica. Dentro de dicha interfaz, podemos realizar varias tareas, como gestionar archivos, directorios, ejecutar programas, diagnosticar problemas del sistema, etc. [31].

  Principales características:
  - Navegar entre directorios podemos utilizar comandos como "**cd**" para cambiar entre ellos.
  - Gestionar archivos: Copiar ("**copy**"), mover ("**move**"), eliminar ("**del**") y renombrar ("**ren**").
  - Ejecutar programas: Lanzar aplicaciones y scripts directamente desde la línea de comandos.
  - Consultar información del sistema: Obtener detalles sobre el sistema y la red con comandos como "**ipconfig**", "**systeminfo**", y "**tasklist**".

- **PowerShell**
  Es una solución de automatización de tareas multiplataforma compuesta por un shell de línea de comandos, un lenguaje de scripting y un marco de gestión de configuración. PowerShell se ejecuta en Windows, Linux y macOS. A diferencia de la mayoría de los shells que solo aceptan y devuelven texto, PowerShell acepta y devuelve objetos .NET.
  Windows PowerShell es un shell de línea de comandos basado en tareas y un lenguaje de scripting diseñado especialmente para la administración del sistema. Desarrollado sobre .NET Framework, Windows PowerShell ayuda a los profesionales de TI y a los usuarios avanzados a controlar y automatizar la administración del sistema operativo Windows y de las aplicaciones que se ejecutan en Windows [25].

  PowerShell sirve como lenguaje de script, se utiliza habitualmente para automatizar la gestión de sistemas. También se utiliza para crear, probar e implementar soluciones, a menudo en entornos de CI/CD. Está basado en Common Language Runtime (CLR) de .NET. Todas las entradas y salidas son objetos .NET. No es necesario analizar la salida de texto para extraer información de la salida [26].

  Principales características:
  - Historial de línea de comandos robusto.
  - Completado de tabulaciones y predicción de comandos (ver about_PSReadLine).
  - Admite alias de comandos y parámetros.
  - Canalización para encadenar comandos.
  - Sistema de ayuda en consola, similar a las páginas de Unix.

- **RDP** Windows
  El protocolo Escritorio remoto de Microsoft (RDP) proporciona funcionalidades de entrada y visualización remota a través de conexiones de red para aplicaciones basadas en Windows que se ejecutan en un servidor, a su vez, está diseñado para admitir diferentes tipos de topologías de red y varios protocolos LAN, se encuentra encapsulado y cifrado dentro de TCP. Se basa en y es una extensión de la familia de estándares de protocolo T-120. Es un protocolo compatible con varios canales incluidos los canales virtuales independientes que pueden llevar información como: datos de presentación, comunicación de dispositivo serie, Información de licencias, datos altamente cifrados, como el teclado, la actividad del mouse, etc [21].

  Principales características [22]:
  - Cifrado: RDP usa el cifrado RC4 de RSA Security, un cifrado de flujo diseñado para cifrar de forma eficaz pequeñas cantidades de datos.
  - Reducción de ancho de banda: RDP admite varios mecanismos para reducir la cantidad de datos transmitidos a través de una conexión de red. Los mecanismos incluyen la compresión de datos, el almacenamiento en caché persistente de mapas de bits y el almacenamiento en caché de glifos y fragmentos en RAM.
  - Teledirigido: El personal de soporte técnico de equipos puede ver y controlar una sesión de escritorio remoto. Compartir la entrada y mostrar gráficos entre dos sesiones de Escritorio remoto ofrece a una persona de soporte técnico la capacidad de diagnosticar y resolver problemas de forma remota.
  - Canales virtuales: Mediante el uso de la arquitectura de canal virtual RDP, se pueden aumentar las aplicaciones existentes y se pueden desarrollar nuevas aplicaciones para agregar características que requieren comunicaciones entre el dispositivo cliente y una aplicación que se ejecuta en una sesión de escritorio remoto.
  - Sonido, unidad, puerto y redirección de impresora de red: Los sonidos que se producen en el equipo remoto se pueden escuchar en el equipo cliente que ejecuta el cliente RDP y las unidades de cliente locales estarán visibles para la sesión de escritorio remoto.
  - Autenticación de tarjeta inteligente a través de Servicios de Escritorio remoto.

- **Visual Studio Code**
  Es un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Es software gratuito y de código abierto, aunque la descarga oficial está bajo software privativo e incluye características personalizadas por Microsoft [41].

  Principales características de Visual Studio Code:

- Interfaz de Usuario Minimalista: Ofrece una interfaz limpia y moderna que puede ser personalizada según las preferencias del usuario. Visual Studio Code ofrece una interfaz de usuario moderna y limpia, que puede personalizarse completamente mediante temas, iconos y disposición de paneles.
- Amplio soporte de extensiones: VS Code permite la instalación de miles de extensiones desde su marketplace para añadir compatibilidad con nuevos lenguajes, depuradores, linters, temas y herramientas de desarrollo.
- Autocompletado inteligente (IntelliSense): IntelliSense proporciona sugerencias de código, autocompletado y documentación contextual para una gran cantidad de lenguajes, como Python, JavaScript, C++, etc.
- Depuración integrada: El editor cuenta con un sistema de depuración visual para múltiples lenguajes. Puedes establecer puntos de interrupción, inspeccionar variables y controlar la ejecución del programa desde el mismo entorno.
- Control de versiones integrado (Git): VS Code incluye herramientas para trabajar con Git sin salir del editor: realizar commits, ver diferencias, gestionar ramas y resolver conflictos.
- Terminal integrada: El editor cuenta con una terminal incorporada (bash, PowerShell, ssh, etc.), lo cual permite ejecutar comandos sin cambiar de ventana o aplicación.
- Soporte multiplataforma y trabajo remoto: VS Code funciona en Windows, macOS y Linux. También permite trabajar de forma remota mediante SSH, contenedores Docker y WSL (Windows Subsystem for Linux).

- **Sublime-Text**
  Es un editor de texto y código fuente muy utilizado debido a su velocidad, versatilidad y una interfaz de usuario muy intuitiva. Es una herramienta poderosa que puede adaptarse a diferentes flujos de trabajo y estilos de programación [32].

  Principales características:
  - Interfaz de Usuario Minimalista: Ofrece una interfaz limpia y moderna que puede ser personalizada según las preferencias del usuario.
  - Alto rendimiento: Conocido por su velocidad y eficiencia, incluso cuando se trabaja con archivos grandes o proyectos extensos.
  - Multiplataforma: Disponible para Windows, macOS y Linux.
  - Soporte para Múltiples Lenguajes: Compatible con una amplia variedad de lenguajes de programación y marcado, incluyendo HTML, CSS, JavaScript, Python, Ruby, y más.
  - Sublime Text Command Palette: Permite acceder rápidamente a comandos, funciones y configuraciones mediante una interfaz de búsqueda rápida.
  - Funciones de Edición Avanzada: Incluye características como edición simultánea, selección múltiple, y búsqueda y reemplazo avanzados.
  - Plugins y Paquetes: Extensible a través de una gran cantidad de plugins y paquetes disponibles en el Package Control, lo que permite añadir nuevas funcionalidades y mejorar el flujo de trabajo.

- Sublime Text Snippets: Permite crear y usar fragmentos de código reutilizables, lo que acelera el desarrollo y la escritura de código.
- Configuración Personalizable: Los usuarios pueden ajustar la apariencia y el comportamiento del editor a través de archivos de configuración en formato JSON.

- **Python [23 - 24]**
  Es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Python es altamente utilizado por su eficiencia y alta versatilidad en su ejecución, siendo compatible con varias plataformas, se puede descargar gratis desde internet y se integra bien a todos los tipos de sistemas, aumentando la velocidad de desarrollo.

  Principales características:
  - Lenguaje interpretado: Python es un lenguaje interpretado, lo que significa que ejecuta directamente el código línea por línea. Si existen errores en el código del programa, su ejecución se detiene, por lo que la velocidad de detección de errores aumenta.
  - Lenguaje fácil de utilizar: Python al igual que algunos otros lenguajes de programación, utiliza palabras iguales y similares a las del inglés. Por lo que su comprensión sintáctica se facilita.
  - Lenguaje tipado dinámicamente: Dentro de Python, no se tienen que anunciar los tipos de variables dado que los determina en el tiempo de ejecución.
  - Lenguaje orientado a objetos: Python considera todo como un objeto, pero también admite otros tipos de programación, como la programación estructurada y la funcional.
  - Código abierto: Python se desarrolla bajo una licencia de código abierto aprobada por OSI, lo que lo hace de libre uso y distribución, incluso para uso comercial. La licencia de Python está administrada por la Python Software Foundation.
  - Aplicaciones: El índice de paquetes de Python (PyPI) alberga miles de módulos de terceros para Python. Tanto la biblioteca estándar de Python como los módulos aportados por la comunidad permiten infinitas posibilidades.

- **Node.js [45]**
  Es un entorno de ejecución que permite ejecutar JavaScript en el servidor, utilizando el motor V8 de Google Chrome. Node.js nos permite utilizar JavaScript en el backend, por lo que facilita el desarrollo full-stack utilizando el mismo lenguaje tanto para el frontend como para el backend, con esto podemos manejar solicitudes HTTP, interactuar con bases de datos, leer y escribir archivos, y realizar otras tareas típicas del backend, todo usando el mismo lenguaje.

  Principales características:

- Asíncrono y No Bloqueante: Utiliza un modelo asíncrono y para manejar las respuestas cuando están listas utiliza un sistema de callbacks por lo que jamás bloquea el hilo principal mientras espera respuestas de operaciones como la lectura de archivos, consultas a bases de datos, etc. Esto lo hace muy eficiente y adecuado para aplicaciones con alto volumen de tráfico concurrente, ya que puede manejar muchas operaciones al mismo tiempo sin quedarse bloqueado esperando resultados.
- Basado en un único hilo (Single-Threaded): Node.js funciona en un solo hilo, aprovechando el bucle de eventos. El event loop nos permite que las operaciones asíncronas se gestionen de manera eficiente y que el código continúe ejecutándose sin esperar a que se completen todas las tareas I/O.
- V8 Engine: Node.js utiliza el motor V8 de Google Chrome, que convierte el código JavaScript en código máquina, esto nos permite aplicaciones optimizadas y de alto rendimiento.
- NPM (Node Package Manager): Este es un gestor de paquetes que permite gestionar bibliotecas y dependencias fácilmente. NPM cuenta con una vasta cantidad de paquetes y módulos listos para ser usados lo que permite la integración de funcionalidades de manera rápida.
- Escalabilidad y rendimiento: La naturaleza de non-blocking I/O nos permite la escalabilidad en las aplicaciones, permitiendo manejar miles de conexiones simultáneas sin comprometer el rendimiento. A su vez permite escalar las aplicaciones agregando más instancias de Node.js o distribuyendo la carga entre varios servidores.
- Manejo de eventos: Está basado en un modelo de eventos, lo que permite que los eventos se disparen cuando una tarea asíncrona se completó, esto mediante un event loop que asegura que las funciones de callback se ejecuten cuando la operación haya terminado.
- Desarrollo en tiempo real: Es un entorno orientado a las aplicaciones en tiempo real, como chats, juegos en línea, o cualquier aplicación que requiera actualizaciones rápidas y constantes entre el servidor y el cliente. La naturaleza asíncrona y el uso de WebSockets hacen que sea sencillo manejar interacciones en tiempo real.

- **Express [46]**
  Este es un framework minimalista y flexible para construir aplicaciones web y APIs con Node.js. Express.js proporciona funcionalidades para crear servidores HTTP de forma sencilla, gestionando rutas, peticiones, respuestas, middlewares, etc. Está diseñado para construir tanto aplicaciones web completas como APIs RESTful.

  Principales características [34]:
  - Minimalista y flexible: No impone una estructura rígida, lo que permite organizar el código como se prefiera.
  - Middleware: Usa funciones middleware que tienen acceso al objeto de solicitud (req), respuesta (res) y a la función next() para procesar peticiones de forma modular.

- Routing: Su sistema de enrutamiento permite definir rutas con diferentes métodos HTTP (GET, POST, PUT, DELETE, etc.).
- Manejo de errores: Permite definir middlewares de manejo de errores personalizados.
- Integración con bases de datos: Aunque no incluye ORM propio, se integra fácilmente con bibliotecas como Mongoose o Sequelize.
- Desarrollo rápido de APIs REST: Es ideal para construir APIs escalables por su simplicidad y soporte nativo para JSON.

- **React [47]**
  Esta es una biblioteca (library) de JavaScript desarrollada por Meta para construir interfaces de usuario (UI), especialmente para aplicaciones web de una sola página (Single Page Applications - SPA). Esta biblioteca nos permite construir componentes reutilizables que gestionan su propio estado, y que se renderizan mediante un sistema llamado DOM virtual. Es una biblioteca declarativa, basada en componentes, que permite crear interfaces interactivas de forma sencilla. A penas de no ser un framework completo, se puede extender con librerías adicionales como React Router o Redux para desarrollar aplicaciones más complejas.

  Principales características:
  - Basado en componentes: Toda la interfaz de usuario se construye dividiendo la aplicación en componentes reutilizables e independientes.
  - JSX (JavaScript XML): Una extensión de sintaxis que permite escribir HTML dentro de JavaScript, lo que mejora la legibilidad y la estructura del código.
  - DOM virtual: React crea una representación virtual del DOM real, permitiendo actualizar solo las partes necesarias de la interfaz, lo que mejora el rendimiento.
  - Unidireccionalidad de datos: El flujo de datos va de padres a hijos, con lo que se facilita el control del estado de la aplicación.
  - Declarativo: Permite describir cómo debería lucir la UI para un estado determinado, y React se encargará de renderizarla.
  - Ecosistema amplio: Puede integrarse fácilmente con otras bibliotecas para manejo de rutas, estado global, o testing.

- **SQL [48]**
  Es un lenguaje de programación diseñado específicamente para gestionar y manipular bases de datos relacionales. SQL (Structured Query Language) nos permite realizar operaciones como la creación de estructuras de datos, inserción, modificación, eliminación y consulta de datos almacenados en bases de datos. Este es el lenguaje estándar utilizado para interactuar con sistemas de gestión de bases de datos relacionales (RDBMS) como MySQL, PostgreSQL, Microsoft SQL Server, Oracle, entre otros.

  Principales características:
  - Lenguaje declarativo: El usuario indica qué quiere obtener, sin especificar cómo hacerlo.

  - Gestión de datos estructurados: SQL trabaja con datos organizados en tablas con relaciones definidas entre ellas.
  - Subdivisión por funciones:
    - DDL (data definition language): define la estructura de la base de datos (ej. create, alter, drop).
    - DML (data manipulation language): manipula datos (ej. insert, update, delete).
    - DQL (data query language): consulta datos (ej. select).
    - DCL (data control language): controla permisos (ej. grant, revoke).
    - TCL (transaction control language): control de transacciones (ej. commit, rollback).
  - Independencia de plataforma: SQL es un estándar soportado por muchos sistemas gestores, aunque con variaciones específicas.
  - Potente y flexible: Permite realizar consultas complejas mediante cláusulas cómo JOIN, GROUP BY, ORDER BY, HAVING, entre otras.
  - Control de transacciones: Soporta el manejo de transacciones para asegurar la integridad de los datos (ACID).

- **Microsoft Visio**
  Es una solución innovadora que te ayuda a visualizar flujos de proceso empresariales conectados a datos de características integradas que ofrecen las funciones avanzadas de Microsoft 365 en Visio. Permite crear diagramas de flujo de funciones cruzadas, diagramas de red, organigramas, planos de planta, diseños de ingeniería y más. Usa formas y plantillas modernas de manera fácil e intuitiva con Visio [27].

- **Microsoft Word**
  Es un procesador de textos desarrollado por Microsoft, mismo que es uno de los más utilizados en el mundo, es parte del conjunto de aplicaciones de Microsoft Office, que también incluye programas como Excel y PowerPoint. Muy utilizado en redacción de documentos (creación de cartas, informes, ensayos, etc.), diseño de currículums, presentaciones y propuestas. Microsoft Word es una herramienta versátil y poderosa que se utiliza en una variedad de contextos, desde el hogar hasta el entorno profesional, y su amplio conjunto de características lo convierte en una opción popular para la creación y edición de documentos [33].

- **Microsoft PowerPoint [35]**
  Es una aplicación de software de presentación desarrollada por Microsoft, que forma parte del conjunto de aplicaciones de Microsoft Office. Es muy utilizada para crear presentaciones visuales que pueden incluir texto, imágenes, gráficos, tablas y otros elementos multimedia. Nos permite comunicar ideas de manera efectiva a través de presentaciones visuales organizadas. Las presentaciones pueden ser de tipo empresariales, para reuniones, informes, presentación de proyectos, de educación para desarrollar material didáctico, presentaciones para clases o conferencias. Eventos para preparar diapositivas para charlas, seminarios y eventos públicos, etc.

- **Boto3 AWS**
  Boto3 es el kit de desarrollo de software (SDK) de Amazon Web Services (AWS) para Python, Boto3 facilita la integración de su aplicación, biblioteca o script de Python con los servicios de AWS, incluidos Amazon S3, Amazon EC2, Amazon DynamoDB y más. El SDK proporciona una API orientada a objetos, así como acceso de bajo nivel a los servicios de AWS [28].

  Principales características [29,30]:
  - API de recursos: Boto3 dispone de dos niveles distintos de API. Las API de cliente (o de "bajo nivel") proporcionan asignaciones individuales a las operaciones HTTP subyacentes de las API. Las API de recursos ocultan llamadas a la red específicas, pero en vez de ello proporcionan recopilaciones y objetos de recursos para acceder a atributos y realizar acciones.
  - Interfaz uniforme y actualizada: Las interfaces de "cliente" y "recursos" de Boto3 poseen clases generadas dinámicamente dependientes de modelos JSON que describen las API de AWS. Esto nos permite proporcionar actualizaciones muy rápidas con gran uniformidad en todos los servicios soportados.
  - Compatibilidad con Python 2 y 3: Boto3 se diseñó desde el principio para proporcionar compatibilidad nativa con las versiones 2.7+ y 3.4+ de Python.
  - Tareas en espera: Boto3 incorpora "tareas en espera", que sondean automáticamente cambios en los estados predefinidos de los recursos de AWS. Boto3 posee tareas en espera para las API de cliente y de recursos.
  - Características de alto nivel específicas de servicios: Boto3 incorpora numerosas características específicas de servicios, como las transferencias multiparte automáticas para Amazon S3 y las condiciones de consultas simplificadas para Amazon DynamoDB.

- **TLS**
  TLS significa Transport Layer Security y es un protocolo criptográfico diseñado para proporcionar seguridad en las comunicaciones a través de redes inseguras como Internet, su propósito es proteger la transmisión de datos sensibles a través del cifrado y otros mecanismos de seguridad. TLS es el sucesor de SSL (Secure Sockets Layer), y ofrece mejoras significativas en seguridad, rendimiento y eficiencia [51].
  TLS se usa comúnmente en protocolos como HTTPS, SMTP, FTPS, VPNs, VoIP, entre otros y sus funciones principales son garantizar:
  • La confidencialidad de los datos (evita que sean leídos por terceros),
  • La integridad (verifica que no hayan sido alterados),
  • La autenticación (confirma la identidad de los participantes en la comunicación).

  Principales características [29,30]:
  - Cifrado de extremo a extremo: TLS cifra los datos que se transmiten entre cliente y servidor, impidiendo que puedan ser leídos por terceros.

- Autenticación: Usa certificados digitales (X.509) para autenticar a las partes, normalmente al servidor, y a veces al cliente.
- Integridad: Utiliza funciones hash (como SHA-256) para asegurarse de que los mensajes no han sido alterados.
- Handshake: Negociación de parámetros criptográficos, durante esta fase inicial, cliente y servidor acuerdan los algoritmos de cifrado y establecen una clave secreta compartida.
- Reanudación de sesión: Permite reutilizar sesiones anteriores para ahorrar tiempo y recursos.
- Compatibilidad con varios protocolos: TLS es agnóstico del protocolo de aplicación, lo que significa que puede proteger HTTP, SMTP, FTP, etc.
- TLS 1.3: más rápido y seguro Elimina algoritmos obsoletos, reduce la latencia (menos rondas de handshake) y mejora la seguridad general.

Algoritmo del TLS:

1. Inicio del Handshake:
    a. El cliente envía una lista de algoritmos soportados.
    b. El servidor responde con su certificado digital y el algoritmo elegido.
2. Intercambio de claves:
    a. Se establece una clave secreta compartida (por ejemplo, usando Diffie-Hellman).
3. Cifrado de datos:
    a. Una vez establecida la clave, los datos se cifran usando cifrado simétrico (como AES).
4. Verificación de integridad:
    a. Cada mensaje lleva un hash que permite detectar si ha sido alterado.
5. Cierre de sesión seguro:
    a. Cliente y servidor intercambian mensajes de cierre para finalizar la sesión.

5. Diseño

Dentro de esta sección podemos encontrar el diseño de cada uno de los componentes que integran al sistema de orquestación. La representación de los componentes del sistema se lleva a cabo mediante diagramas UML, mismos que describen de manera formal la especificación del flujo del sistema.

5.1. Modelado de la Arquitectura y Despliegue

5.1.1. Diagrama UML de Componentes

El presente diagrama UML de Componentes nos muestra la modularidad del software y las interfaces que lo integran. Dentro de dicho diagrama, se muestra a detalle cada una de las

partes que forman parte del servidor bastión, a su vez muestra cómo se realiza la conexión hace los servidores de producción de los negocios de los clientes. También tiene integrado un programa de monitoreo mismo que sensa cada uno de los servidores de los clientes de forma permanente, guardando cada uno de los cambios que se registran dentro de los mismos. Mientras los servidores permanecen en el mismo estado, este no se guarda, dado que dicho estado ya se encuentra guardado en la base de datos.

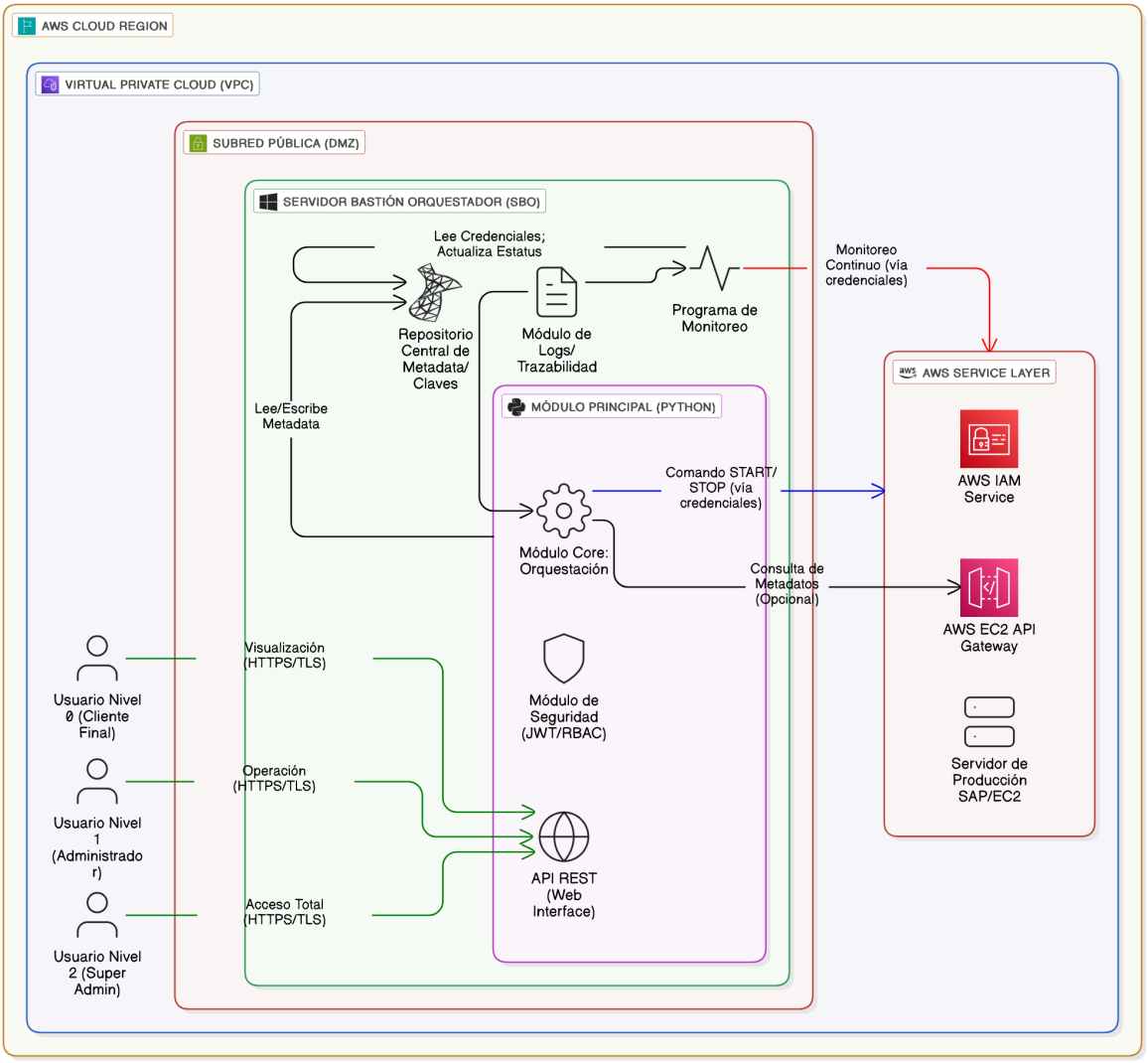


Fig 11. Imagen del diagrama de despliegue del servidor bastión orquestador. Diseño propio.

En el anterior diagrama en la figura 11, se modeló la distribución física y lógica de los componentes de hardware y software en el entorno de AWS. Su propósito es mostrar cómo los componentes de software (SBO) están instalados y se comunican a través de la infraestructura de nube (Nodos y Conexiones).

5.1.2. Diagrama de flujo del sistema

En el siguiente diagrama se muestra el flujo que sigue el sistema además de los componentes que lo integran. Dentro de los principales componentes se encuentra una base de datos contenida dentro del servidor bastión, misma que contiene todos los metadatos de los clientes y los negocios de los mismos. También contiene el módulo de orquestación central que recupera las credenciales del usuario que esté utilizando el sistema y de acuerdo con el nivel del mismo, la permite ciertos privilegios de operación o solo la visualización. En la esquina inferior derecha de la fig 12, también podemos encontrar la subrutina que se lleva a cabo dentro de un servidor de producción si es que llega una petición de encendido o apagado y el usuario tenía los privilegios para operarlo.

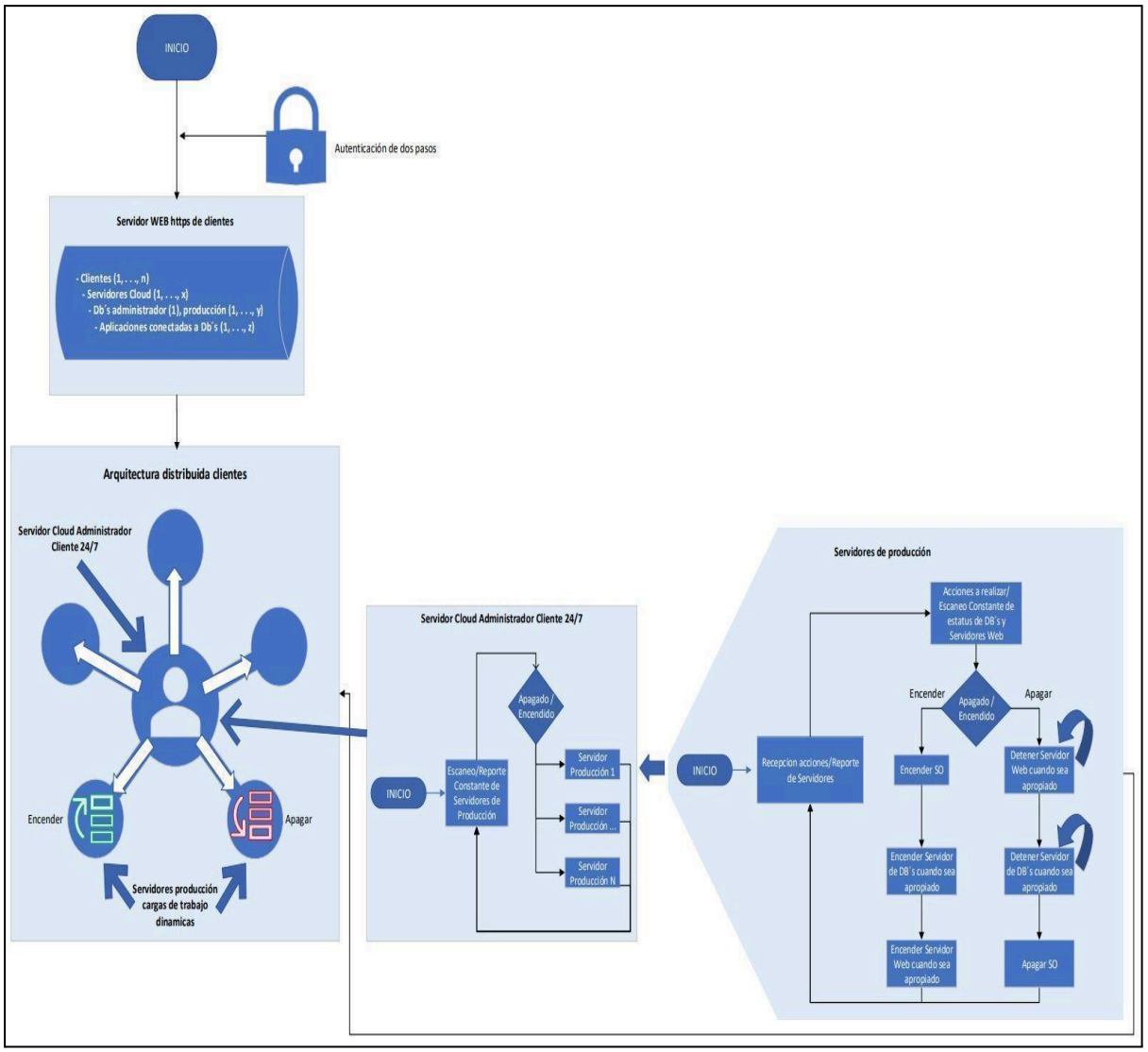


Fig 12. Diagrama de flujo del sistema. Diseño propio.

5.2. Componentes del sistema

En esta sección se lleva a cabo la descripción de cada uno de los componentes que integran el sistema del orquestador.

5.2.1. Seguridad

5.2.1.1. Autenticación y control de acceso

Implementar Autenticación Multifactor (MFA) utilizando AWS Cognito, para garantizar que únicamente usuarios autorizados en este caso los administradores designados dentro del sitio web puedan acceder a la plataforma y puedan administrar el estatus de cada uno de los servidores. Con esto podemos llevar a cabo el registro de los usuarios deseados y hacer que a cada uno de ellos les llegue una contraseña temporal a su correo o celular según se defina para poder acceder a la plataforma de operación.

En la primera parte del diagrama, se muestra la autenticación la cual consta de dos pasos, con lo cual se propone invalidar el acceso no autorizado a la plataforma a personal no registrado en la misma. Además de la autenticación de dos pasos, se propone utilizar conexiones seguras, es decir utilizar protocolos https, para que la conexión vaya cifrada entre el administrador de la plataforma y la misma.

5.2.1.2. Cifrado de Datos

Implementar cifrado de datos en tránsito utilizando SSL/TLS para todas las conexiones web. Esto para proteger cada una de las conexiones hacia el servidor central. Para esto es necesario trabajar con los certificados de autenticación de cada una de las partes, se propone trabajar sobre ECDSA y manejar los niveles más altos de seguridad disponibles.
A su vez se está considerando el utilizar grupos de seguridad (Security Groups) y Listas de Control de Acceso a la Red (NACLs) para controlar el acceso al servidor central o utilizar Firewalls y VPN para garantizar que las comunicaciones sean seguras y controladas. Aunque se tendrían que valorar más a profundidad los pros y contras de la implementación de estas políticas dado que nos limitaría en la movilidad de los administradores. Mientras ellos se encuentren en sus puestos de trabajo no habría problema en la apertura de una conexión vpn permitiendo la conexión hacia el servidor central y de ahí permitir la manipulación de los servidores de los clientes.

5.2.2. Clientes

La plataforma contempla el registro de clientes, junto a las arquitecturas en la nube que dichos clientes tengan operando en la nube. Se hace hincapié en que por el momento todos los datos con los que trabajaremos serán datos lo más apegado a la realidad, pero no

podrán ser reales, dado que el tipo de información que los servidores SAP contienen y procesan es muy sensible.

5.2.2.1. Servidor web https de los clientes

Dicho servidor contendrá una lista de clientes hipotéticos junto a las direcciones y claves asociadas a cada uno de los servidores dentro de las arquitecturas desplegadas en los servidores de AWS en donde se encuentran operando los sistemas SAP asociados a las operaciones de las empresas de los clientes.

5.2.2.2. Nubes públicas

Se contempla el registro de la nube pública sobre la cual los clientes tienen a sus servidores de producción en renta, dado que derivado de cada una de ellas las formas de conexión variarán de una a otra. Para el presente trabajo se contempla únicamente el registro de las nubes, aunque no la operatividad dentro de cada una de ellas, limitándose a la nube de AWS por parte de la empresa Amazon.

5.2.3. Arquitectura de los clientes

Las arquitecturas utilizadas por cada uno de los clientes registrados dentro de la plataforma web, podrá variar, entre cada uno de ellos e inclusive entre cada uno de los negocios asociados a un cliente en específico, dichas arquitecturas se describen más adelante en el apartado “arquitectura de un servidor de producción”.
Dicho lo anterior, la cantidad de negocios asociados a un cliente y la cantidad de servidores asociados a dichos negocios puede ser variable entre cada uno de ellos, pudiendo ser diferentes los sistemas operativos, gestores de bases de datos y más. Aunque en la presente propuesta solo se contempla la operación sobre servidores registrados sobre la nube de AWS, los sistemas operativos Windows Server, Linux y los gestores de bases de datos: Microsoft Sql Server y Oracle.

5.2.3.1. Servidores de producción

Se contempla el registro de todos los datos de conexión de cada uno de los servidores de los clientes junto a las claves asociadas a cada uno de ellos, para poder en el futuro acceder y poder operar de forma remota cada uno de los servidores que algún cliente requiera encender o apagar.
Dependiendo del cambio de estatus que se quiera llevar a cabo sobre un servidor este debe seguir en general ciertos pasos para realizar su cambio de estatus (encender/apagar). Una vez que un determinado servidor reciba la orden del cambio de estatus, este debe comenzar a ejecutarla previa revisión de la autenticación asociada a cada uno de ellos.

5.2.3.2. Arquitectura de un servidor de producción

En la figura 12, se muestra la heterogeneidad que podría tener cada uno de los servidores de producción asociados a la arquitectura registrada por un cliente [19].

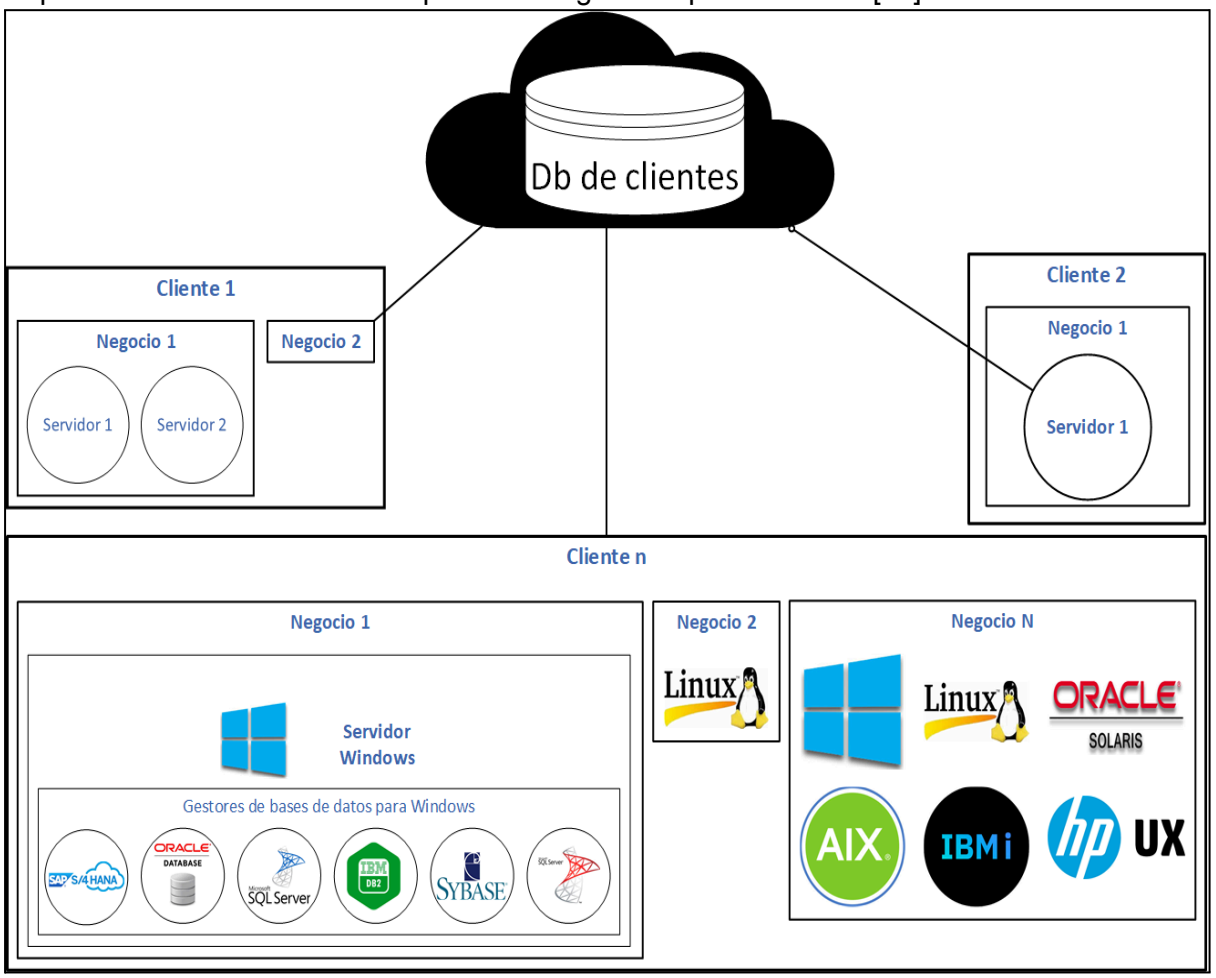


Fig 13. Diagrama de un servidor de producción. Diseño propio.

Cliente 1:
En la esquina superior derecha de la figura 13, podemos ver la descripción abstracta de la arquitectura asociada al cliente 1, en la cual, podemos ver que tiene 2 negocios, en el primer, tiene registrados 2 servidores de producción dentro de nuestra plataforma, no especificando el sistema operativo asociado a cada uno de ellos. A su vez podemos ver que en el negocio 2, podría tener o no un servidor en la nube o tal vez no, por lo que no se especifica, dado que esto no es un problema que se deba considerar en el presente esquema. Ambos servidores permanecen registrados junto a un negocio perteneciente a un cliente en específico.

Cliente 2:
Podemos ver que el cliente 2 tiene registrado dentro de la plataforma únicamente un negocio y dentro dicho negocio un servidor de producción, no se especifica si dicho cliente tiene más negocio o servidores en la nube, sin embargo, para nuestro propósito, no hay problema, dado que se podrá operar dicho servidor toda decisión que el cliente tome sobre

él. También podemos apreciar en el esquema, que el tamaño del servidor en cuestión tiene un tamaño diferente a los tamaños descritos con el anterior cliente, mostrando que las capacidades asociadas a cada servidor pueden cambiar, dado los requerimientos en el negocio de cada cliente. Una vez más, no se especifica el SO, gestor de base de datos, motor web o más detalles asociados al mismo.

Cliente n, negocio 1:
A su vez podemos ver en la parte inferior de la figura 12, se nos muestra un cliente 'n', el cual cuenta con más de un negocio, podemos observar que con este cliente no se muestra la cantidad de servidores asociados a cada uno de los negocios del mismo, sin embargo se nos muestra que en el primer servidor tiene registrado uno o más servidores con el sistema operativo "Windows Server", dicho servidor/servidores podrían tener asociados alguno de los gestores de bases de datos que a continuación se enlistan.

Cliente n, negocio 2:
Como en el caso anterior, en el negocio 2 no se nos muestra la cantidad de servidores asociados al mismo, sin embargo, se nos muestra que los sistemas operativos asociados a dicho negocio son "Linux" mostrándonos que dependiendo del giro del negocio de un cliente los sistemas operativos asociados a los servidores de producción de dichos negocios podrían variar según las conveniencias de cada cliente.

Cliente n, negocio n:
Para este negocio se nos muestra que los sistemas operativos con los que es compatible el sistema SAP, pueden ser varios, en este caso ya no se nos muestra los motores de bases de datos asociados a cada uno de los sistemas operativos, aunque en la descripción del negocio 1 si se nos mostró la compatibilidad de gestores de bases de datos compatible con el sistema operativo Windows.

Sistemas operativos compatibles con sistema SAP:
- Windows
- IBM i
- Unix AIX
- Unix HP-Aux
- Unix Linux
- Unix Oracle Solaris

5.2.3.3. Gestores de bases de datos

Cada uno de los servidores contiene un gestor de base de datos operando dentro de sus arquitecturas, dichos datos del gestor de la base de datos junto a las claves de conexión hacia las mismas son importantes dado que se requieren para administrar las bases de datos contenidas en cada uno de los servidores de cada cliente registrado en la plataforma.

Gestores de bases de datos compatibles con SO Windows:
- SAP MaxDB
- SAP HANA Database
- SAP Adaptive Server Enterprise
- Oracle
- MS SQL Server
- IBM Db2 for z/OS
- IBM Db2 for Windows
- IBM Db2 for i

5.2.4. Servidor Bastión (Orquestador)

La figura del "orquestador", dependiendo del contexto en el que nos encontremos trabajando, puede cambiar su significado. Dentro de la gestión de proyectos, se refiere a un gerente de proyecto que supervisa el avance general de cada una de las partes que lo componen. También puede ser un gestor que coordina procesos y recursos dentro de una organización para garantizar que todas las partes trabajen juntas de manera efectiva para alcanzar diferentes objetivos.
En el contexto tecnológico nos encontramos con una herramienta como “Kubernetes” la cual nos ayuda en la gestión de contenedores dado que administra contenedores y microservicios, actúa como orquestador al coordinar el despliegue y escalado de aplicaciones distribuidas [10].
Por lo cual en la presente propuesta nos referimos al “orquestador” como una herramienta de software que gestionará servidores y servicios, así como aplicaciones para semi-automatizar el encendido y apagado de servidores en la nube.
Como en el apartado anterior se describió, el presente trabajo pretende gestionar y orquestar el funcionamiento de cada uno de los servidores registrados dentro de la plataforma, toda vez que las claves y permisos asociados a cada uno de ellos así lo permitan. El registro de cada uno de los clientes, arquitecturas, servidores, así como las claves asociadas a cada servidor se debe llevar de forma meticulosa dado que un cambio en alguno de los campos derivará en la no operatividad de algún servidor según lo descrito.

El servidor bastión, mismo que contener la lista de clientes hipotéticos registrados dentro de la plataforma, se encontrará escaneando de forma periódica cada uno de los servidores que contenga registrados dentro de sí, esto con la finalidad de en todo momento contener información fidedigna sobre el estado de cada uno de ellos y poder realizar la operación del cambio de estatus en cada uno de los servicios asociados a los mismos.
En cuanto el administrador se conecte al servidor, este podrá revisar el último estatus asociado de cada uno de los servidores, dado que este se va actualizando de forma periódica. A su vez el administrador podrá realizar el cambio de estatus de un servidor toda vez que reciba la instrucción del cliente final para realizar dicho cambio, con lo cual de forma intuitiva y transparente deberá mandar los comandos asociados al cambio de estatus de un servidor en específico de un cliente y negocio en específico pudiendo saber en todo momento el estatus de cada uno de los servicios asociados a la infraestructura de SAP en todo momento.

5.2.4.1. Monitoreo Proactivo

Para el presente trabajo estamos proponiendo el uso de bash scripting para monitorear el rendimiento de cada uno de los servidores de producción con los que estaremos trabajando. Esto sería una especie de sustituto del servicio AWS CloudWatch para monitorear el rendimiento de las instancias EC2 el cual además de monitorear, nos permite diseñar alarmas y realizar acciones automáticas.
En este sentido la idea es que el monitoreo del estatus de cada uno de los servidores de producción asociados al servidor bastión le reporten a este, mediante algún tipo de mensajes acerca de la salud de estos servidores.
En la figura 14, se muestra la propuesta basada en cambios, con lo cual se minimiza la cantidad de mensajes que se envían de los clientes al servidor reduciendo los mensajes de

estados repetitivos. Con lo cual la base de datos refleja únicamente los cambios en el rendimiento de los servidores si es que estos cambiaron en algún sentido.

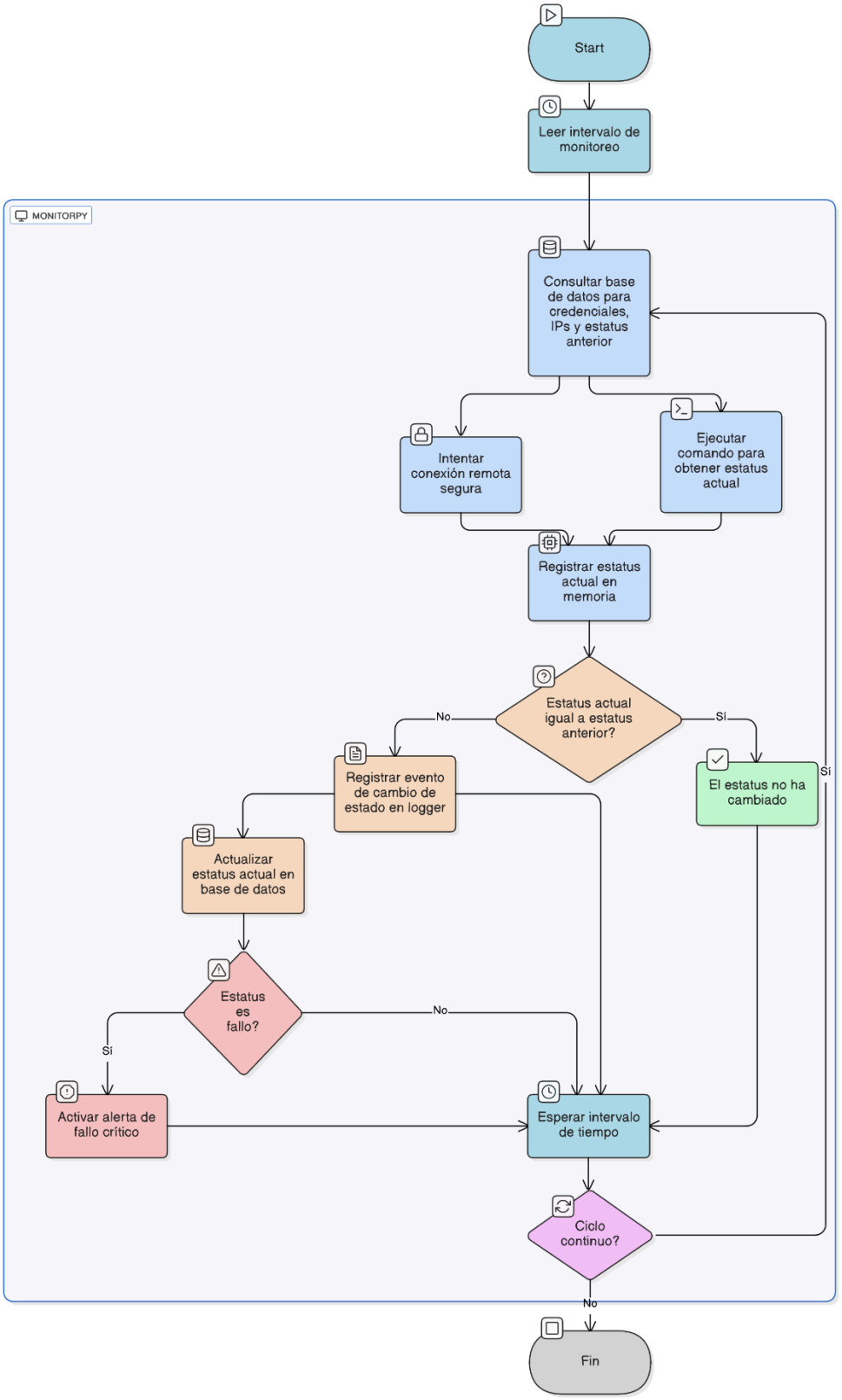


Fig 14. Diagrama del programa monitor que sustituye a Cloudwatch. Diseño propio.

Plataforma Web intuitiva alojada en el servidor bastión

Desarrollar una aplicación web interactiva usando el frameworks React para crear un panel de administración intuitivo. Dicha interfaz debe permitir a los administradores gestionar instancias, monitorear el estado de los servidores y ejecutar comandos. Desde aquí los usuarios definidos como administradores, podrán llevar a cabo las operaciones de administración sobre los servidores de producción que los clientes finales definan. Por lo cual se debe implementar un sistema de notificaciones en tiempo real mediante WebSockets para alertar a los administradores sobre el estado de las instancias si se encuentran encendidas, apagadas o se han detectado algunos fallos.

Escalabilidad Dinámica

Para este trabajo dado que como ya se ha mencionado antes, únicamente se trabajará con 2 servidores de producción, el escalado dinámico no aplica por las cargas pequeñas con las que trabajaremos. Sin embargo, en un entorno real, es necesario estar monitoreando y en dado caso aplicar las políticas de escalado dinámico para distribuir las cargas que reciben los servidores de producción.

5.3. Diagrama de Secuencia UML del sistema

En este apartado podemos observar el diagrama de secuencia que sigue cada una de las partes que componen al sistema. El siguiente diagrama representa el flujo que siguen las peticiones dentro del servidor Bastión mismo que alberga todo el sistema de orquestación.
El presente diagrama formaliza la arquitectura del sistema, especificando los nodos físicos (Windows Server 2019/SQL Server) y la topología de red (Dominio orquestador.top). En él se detalla la interacción temporal y el orden de mensajes para la rutina de apagado (Stop), que es inversa al encendido, crucial para la integridad de datos. Su propósito es asegurar la secuencia crítica inversa: Application → DB → SO antes de apagar las instancias EC2.
Para llegar a este punto, se investigó el proceso manual de encendido/apagado de sistemas SAP, identificando su secuencia crítica para evitar la corrupción de datos. Dicho proceso se modeló de forma que evita la corrupción de datos, también muestra la secuencia de pasos para que el Servidor Bastión Orquestador (SBO) inicie los servicios SAP/EC2.

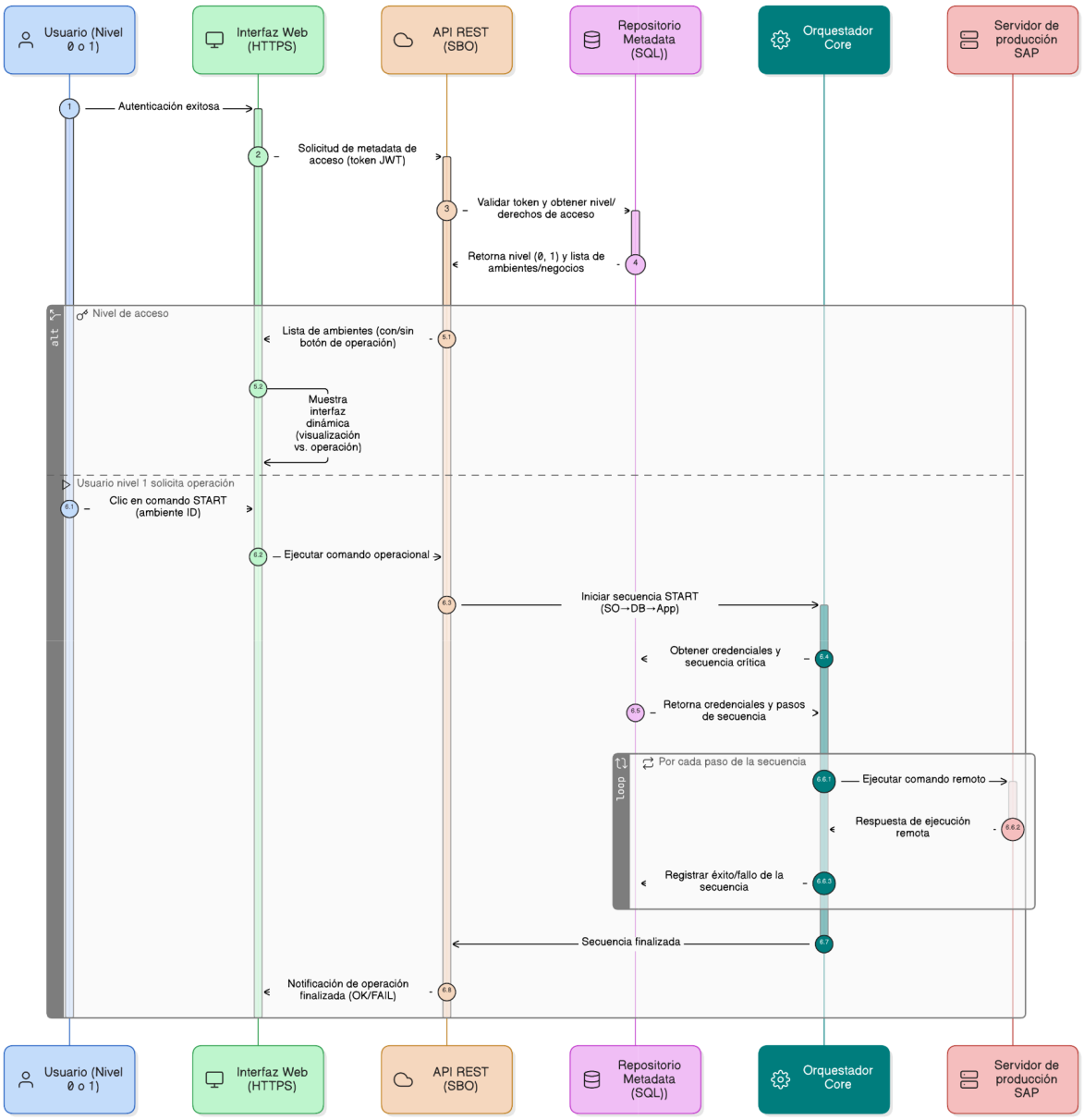


Fig 15. Imagen del diagrama de la interacción y orden de mensajes para la rutina de apagado (Stop). Diseño propio.

El anterior diagrama de Secuencia UML mostró el flujo de operación y control de Interfaz centrándose en la lógica del RBAC (Interfaces Dinámicas) y el proceso de Operación (Start o Stop) a través de la API del SBO hacia los servidores de producción. También se muestra como la API Web consulta los derechos del usuario en turno para presentar la interfaz correcta, esto dependiendo del nivel de usuario que se encuentre utilizando el sistema.

Para llevar a cabo el encendido o apagado de los servidores de producción, se debe seguir una secuencia lógica para no incurrir en alguna corrupción de los datos contenidos en dichos servidores.

Control de Encendido y Apagado de Servidores

Diseñar archivos bash alojados dentro de cada uno de los servidores de producción, que alberguen las rutinas a seguir cuando estos reciban la orden de ejecución. En este sentido, dichas rutinas están diseñadas para llevar a cabo el apagado de cada uno de los servidores de producción en un orden determinado, tomando en cuenta el estatus de los servidores web pertenecientes a la capa de aplicación del sistema ERP SAP, después el estatus de los servidores de bases de datos y por último si cada una de las condiciones se cumplen, llevar a cabo el apagado del servidor de producción en cuestión.

Encendido

El encendido de un servidor conlleva la inicialización de una subrutina implementada en cada uno de los servidores previamente, la cual consta de que se autoinicia de forma automática una vez que cada servidor sea iniciado, con lo cual dicha subrutina, debe reportar el estatus del servidor al servidor Maestro (Bastión), acto seguido, una vez que el sistema operativo se encuentre completamente operativo, la subrutina debe desencadenar el encendido del gestor de base de datos asociado al sistema SAP, toda vez que el estatus del SO lo permita y se encuentre en las condiciones óptimas para operar.
Concluida la fase anterior, se debe encender el servidor web que cada uno de los servidores tenga asociado, con lo cual una vez terminada dicha acción, la operatividad del sistema SAP asociada a dichos servidores quedará operativa y a la espera de las cargas que cada uno de los clientes tenga destinada hacia dichos servidores.

Apagado

El proceso de apagado de cada uno de los servidores se realizará en el sentido inverso del anterior paso, con la diferencia de que cada uno de los comandos asociados a cada una de las acciones anteriormente descritas se ejecutará, cuando las cargas asociadas a cada servidor lo permita, es decir, cada uno de los pasos se tratará de ejecutar dependiendo de la carga que cada servidor tenga en el momento de la recepción del comando de apagado. Dando prioridad al detenido pasivo de cada uno de los servicios y así evitar, que la información contenida en los servidores sufra algún tipo de corrupción, o pérdida de datos.

6. Implementación

6.1. Servidores de producción

Los servidores de producción que se están utilizando para este proyecto, son instancias de AWS del tipo EC2, cada uno de los cuales está asociado a un negocio y cuenta con una base de datos y un servidor web mediante los cuales se simula la interacción de un sistema SAP con los usuarios finales relacionados a un negocio que interactúan con él.

6.1.1. Servidor Linux Ubuntu

Este servidor de producción, simula las transacciones generadas en una ferretería llamada “Ferremart”, las cuales incluyen la compra-venta de productos industriales de diferentes tipos. Una ferretería es un establecimiento comercial que se dedica a la venta de herramientas, materiales y productos relacionados con la construcción, reparación, mantenimiento del hogar, jardinería y otras actividades similares. Su funcionamiento incluye una serie de procesos organizados que permiten ofrecer productos y servicios de manera eficiente [39].
Por lo cual en este servidor contamos con una base de datos que contiene diferentes tablas, cada una de ellas con un propósito diferente. A su vez se desarrolló un programa que simula las transacciones de compra de insumos y venta de productos al público en general.

Base de datos
En esta base de datos se refleja la relación que existe entre los diferentes departamentos dentro de una ferretería. Un producto pertenece a una categoría y tiene un proveedor. Una venta se realiza a un cliente y puede tener una factura. Cada venta contiene varios detalles de venta y una compra se hace a un proveedor y se detalla en detalle_compra.
En la figura 16, podemos distinguir cada una de las tablas que componen la base de datos de este cliente.

Las tablas con las que cuenta la base de datos son las siguientes:

- **“Categorías”**
  Clasifica los productos según su tipo o área (Ej.: Herramientas, Pinturas).
  - **“id_categoria”**: Identificador único.
  - **“nombre_categoria”**: Nombre de la categoría.

- **“Proveedores”**
  Almacena información sobre los proveedores que suministran productos a la ferretería.
  - **“id_proveedor”**: Identificador único.
  - **“nombre”**: Nombre de la empresa o persona proveedora.
  - **“telefono”**
  - **“direccion”**
  - **“email”**: Datos de contacto.

- **“Productos”**
  Contiene el inventario de productos disponibles para la venta.
  - **“id_producto”**: Identificador único.
  - **“nombre”**: Nombre del producto.
  - **“descripcion”**: Información básica del producto.
  - **“precio”**: Precio de venta al público.
  - **“stock”**: Cantidad disponible.
  - **“id_categoria”**: Categoría a la que pertenece (FK).
  - **“id_proveedor”**: Proveedor del producto (FK).

- **“Clientes”**
  Guarda información de los clientes que realizan compras.
  - **“id_cliente”**: Identificador único.
  - **“nombre”**
  - **“telefono”**
  - **“direccion”**
  - **“email”**: Datos del cliente.

- **“Ventas”**
  Registra las operaciones de venta realizadas a clientes.
  - **“id_venta”**: Identificador único.
  - **“fecha”**: Fecha y hora de la venta.
  - **“id_cliente”**: Cliente que realizó la compra (FK).
  - **“total”**: Importe total de la venta.

- **“Detalle_venta”**
  Desglosa los productos vendidos en cada venta.
  - **“id_detalle”**: Identificador único.
  - **“id_venta”**: Referencia a la venta (FK).
  - **“id_producto”**: Producto vendido (FK).
  - **“cantidad”**: Número de unidades vendidas.
  - **“precio_unitario”**: Precio por unidad en el momento de la venta.

- **“Facturas”**
  Documento oficial asociado a una venta.
  - **“id_factura”**: Identificador único.
  - **“numero_factura”**: Código de factura (único).
  - **“fecha_emision”**: Fecha de creación.
  - **“id_venta”**: Referencia a la venta facturada (FK).
  - **“tipo_pago”**: Método de pago (Ej.: Efectivo, Tarjeta).
  - **“estado”**: Estado de la factura (Ej.: Emitida, Pagada, Anulada).

- **“Compras”**
  Registro de compras realizadas a proveedores para abastecer el inventario.
  - **“id_compra”**: Identificador único.
  - **“fecha”**: Fecha y hora de la compra.
  - **“id_proveedor”**: Proveedor al que se compró (FK).
  - **“total”**: Importe total de la compra.

- **“Detalle_compra”**
  Detalla qué productos fueron adquiridos en cada compra.
  - **“id_detalle”**: Identificador único.
  - **“id_compra”**: Referencia a la compra (FK).
  - **“id_producto”**: Producto adquirido (FK).
  - **“cantidad”**: Número de unidades compradas.

- **“precio_unitario”**: Precio pagado por unidad.

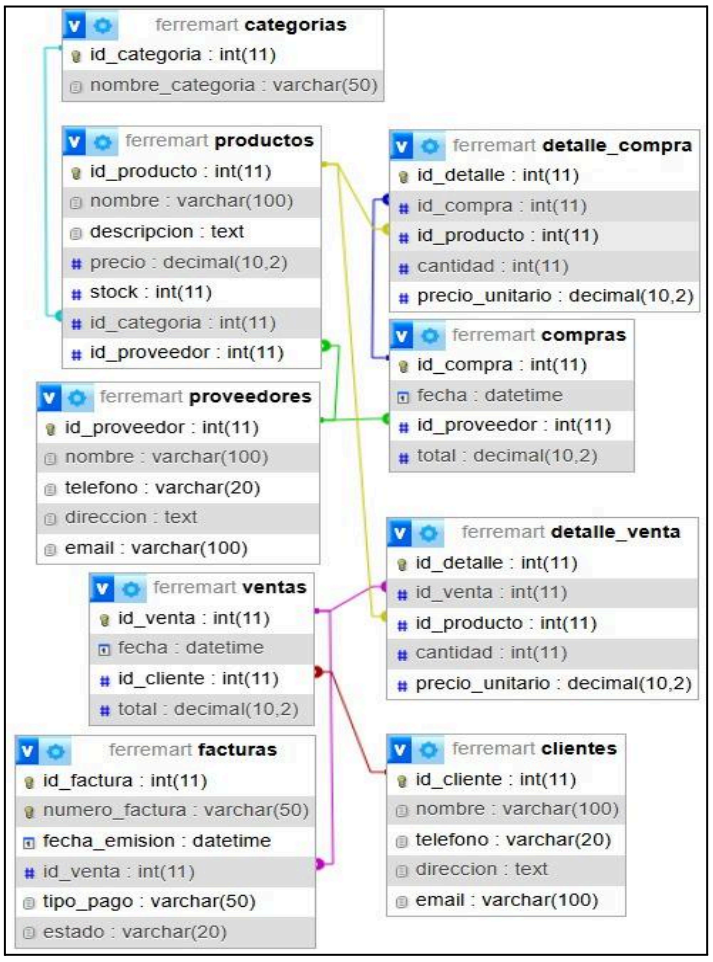
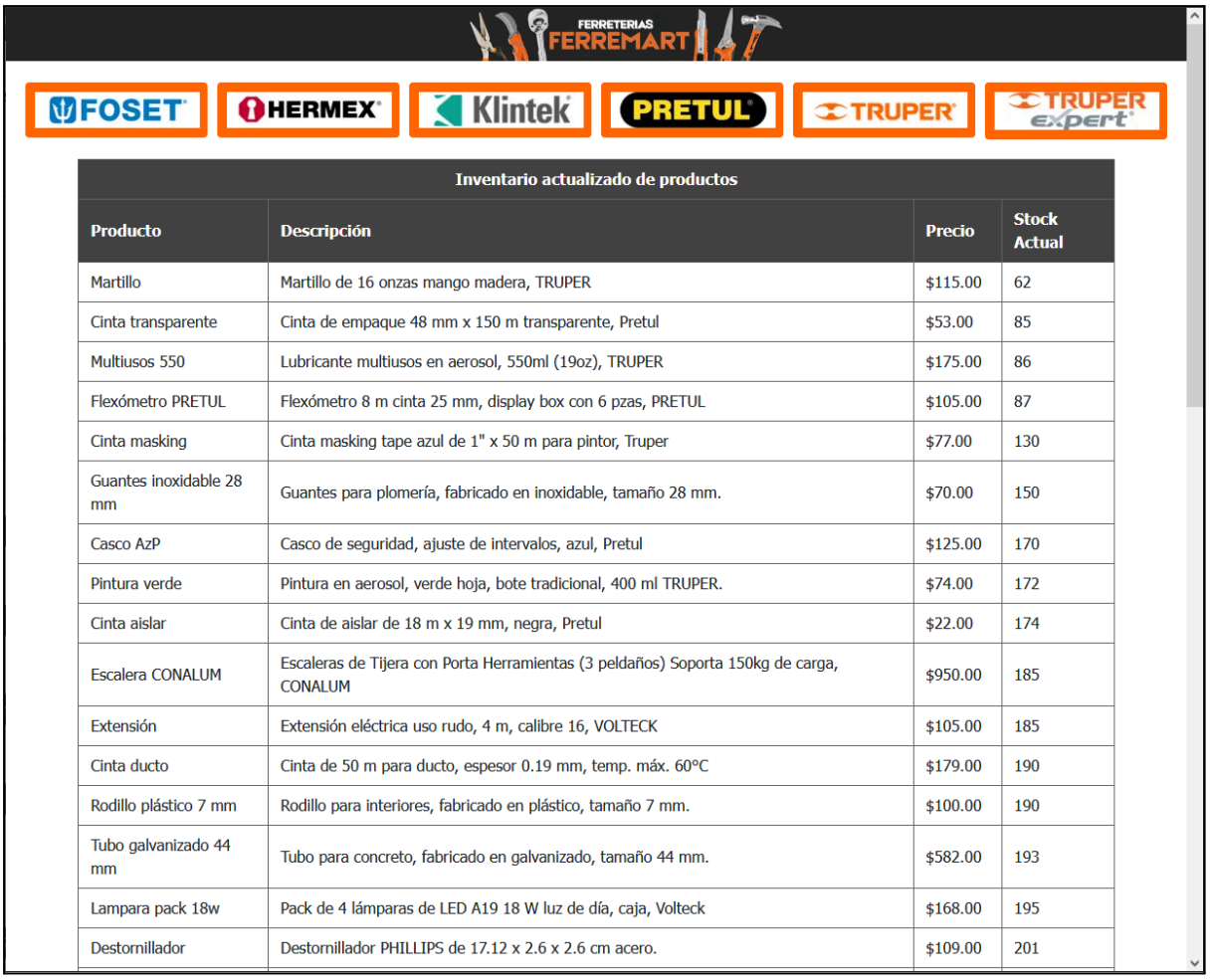


Fig 16. Imagen de la base de datos de la ferretería. Diseño propio.

6.1.1.1. Página web: Ferremart



Inventario actualizado de productos			
Producto	**Descripción**	**Precio**	**Stock Actual**
Martillo	Martillo de 16 onzas mango madera, TRUPER	$115.00	62
Cinta transparente	Cinta de empaque 48 mm x 150 m transparente, Pretul	$53.00	85
Multiusos 550	Lubricante multiusos en aerosol, 550ml (19oz), TRUPER	$175.00	86
Flexómetro PRETUL	Flexómetro 8 m cinta 25 mm, display box con 6 pzas, PRETUL	$105.00	87
Cinta masking	Cinta masking tape azul de 1" x 50 m para pintor, Truper	$77.00	130
Guantes inoxidable 28 mm	Guantes para plomería, fabricado en inoxidable, tamaño 28 mm.	$70.00	150
Casco AzP	Casco de seguridad, ajuste de intervalos, azul, Pretul	$125.00	170
Pintura verde	Pintura en aerosol, verde hoja, bote tradicional, 400 ml TRUPER.	$74.00	172
Cinta aislar	Cinta de aislar de 18 m x 19 mm, negra, Pretul	$22.00	174
Escalera CONALUM	Escaleras de Tijera con Porta Herramientas (3 peldaños) Soporta 150kg de carga, CONALUM	$950.00	185
Extensión	Extensión eléctrica uso rudo, 4 m, calibre 16, VOLTECK	$105.00	185
Cinta ducto	Cinta de 50 m para ducto, espesor 0.19 mm, temp. máx. 60°C	$179.00	190
Rodillo plástico 7 mm	Rodillo para interiores, fabricado en plástico, tamaño 7 mm.	$100.00	190
Tubo galvanizado 44 mm	Tubo para concreto, fabricado en galvanizado, tamaño 44 mm.	$582.00	193
Lampara pack 18w	Pack de 4 lámparas de LED A19 18 W luz de día, caja, Volteck	$168.00	195
Destornillador	Destornillador PHILLIPS de 17.12 x 2.6 x 2.6 cm acero.	$109.00	201

Fig 17. Imagen de la página web con los datos de Ferremart. Diseño propio.

En la figura 17, podemos ver la interfaz principal del cliente Ferremart, aquí podemos distinguir el movimiento de los productos en tiempo real (compras y ventas).

6.1.2. Servidor Windows Server 2019

Este servidor de producción simula las transacciones generadas en una tienda llamada “Oxxo”, las cuales incluyen la compra-venta de productos cotidianos al público en general. Una tienda OXXO es un tipo de tienda de conveniencia que está diseñada para que los clientes puedan comprar rápida y fácilmente productos de uso diario como refrescos, botanas, cigarros, pan, artículos de higiene, así como realizar pagos de servicios (luz, agua, celular), recargas telefónicas además de realizar depósitos bancarios [40].
Este servidor contiene una base de datos que simula la administración de la compra-venta de los diferentes productos dentro de un oxxo. Por ello cuenta con diferentes tablas, cada una de ellas con un propósito diferente. A su vez se desarrolló un programa que simula las

transacciones de compra de productos con los proveedores y venta de productos al público en general.

Base de datos

En esta base de datos se refleja la relación que existe entre los diferentes departamentos dentro de la tienda OXXO. Un producto pertenece a una categoría y tiene un proveedor. Una venta la realiza un cliente y puede tener una factura. Cada venta contiene varios detalles de venta y una compra se hace a un proveedor y se detalla en detalle la compra.
En la figura 18, podemos distinguir cada una de las tablas que componen la base de datos de este cliente.
Las tablas con las que cuenta la base de datos son las siguientes:

- **“Sucursal”**
  Almacena la información de cada tienda OXXO física para saber dónde se realizó una venta, dónde trabaja un empleado o dónde hay inventario.
  - **“id_sucursal”**: Identificador único de la sucursal.
  - **“nombre”**: Nombre asignado a la sucursal.
  - **“direccion”**
  - **“ciudad”**
  - **“estado”**
  - **“codigo_postal”**: Ubicación detallada.

- **“Empleado”**
  Guarda los datos de cada empleado que trabaja en una sucursal para registrar qué empleado hizo una venta y a que sucursal pertenece.
  - **“id_empleado”**: Identificador del empleado.
  - **“nombre”**, apellido: Datos personales.
  - **“puesto”**: Cargo (ej. cajero, gerente).
  - **“salario”**: Sueldo mensual.
  - **“fecha_contratacion”**: Cuándo comenzó a trabajar.
  - **“id_sucursal”**: A qué sucursal está asignado.

- **“Proveedor”**
  Representa a las empresas que suministran productos a la tienda. Permite saber de dónde proviene un producto.
  - **“id_proveedor”**: ID del proveedor.
  - **“nombre”**
  - **“contacto”**
  - **“telefono”**
  - **“direccion”**: Datos de contacto.

- **“Producto”**
  Contiene la información de los artículos disponibles para la venta. Esta tabla está relacionada con inventario, ventas y proveedores.
  - **“id_producto”**: ID del producto.
  - **“nombre”**
  - **“descripcion”**: Nombre y detalles del producto.

  - **“precio”**: Precio de venta.
  - **“categoria”**: Tipo de producto (bebidas, botanas, etc.).
  - **“id_proveedor”**: Quién lo suministra.

- **“Inventario”**
  Controla la cantidad de productos por sucursal sirve para verificar la disponibilidad de productos en cada tienda.
  - **“id_inventario”**: ID único del registro.
  - **“id_sucursal”**: Dónde está el inventario.
  - **“id_producto”**: Qué producto.
  - **“cantidad”**: Cuánto stock hay disponible.

- **“Cliente”**
  Registra información opcional de clientes (para facturación o fidelización).
  - **“id_cliente”**: ID del cliente.
  - **“nombre”**
  - **“apellido”**
  - **“telefono”**
  - **“correo”**: Datos de contacto.

- **“Venta”**
  Representa cada compra realizada por un cliente en una sucursal. Esta tabla está relacionada con clientes, empleados, sucursales y detalles de venta.
  - **“id_venta”**: ID de la venta.
  - **“fecha”**: Cuándo se realizó.
  - **“total”**: Monto total.
  - **“id_sucursal”**
  - **“id_empleado”**
  - **“id_cliente”**: Actores involucrados.

- **“DetalleVenta”**
  Registra los productos específicos incluidos en cada venta, nos sirve para saber qué se vendió, en qué cantidad y a qué precio.
  - **“id_detalle”**: ID del detalle.
  - **“id_venta”**: A qué venta pertenece.
  - **“id_producto”**: Qué producto se vendió.
  - **“cantidad”**: Cuántas unidades.
  - **“precio_unitario”**: Precio por unidad en esa venta.

- **“Factura”**
  Genera un comprobante fiscal de cada venta (para contabilidad o cliente). Es para asociar ventas con un documento fiscal.
  - **“id_factura”**: ID de la factura.
  - **“numero_factura”**: Código único de la factura.
  - **“fecha_emision”**: Cuándo se generó.
  - **“total_facturado”**: Monto facturado.

- **“metodo_pago”**: Forma de pago (efectivo, tarjeta, etc.).
- **“id_venta”**: Venta asociada.

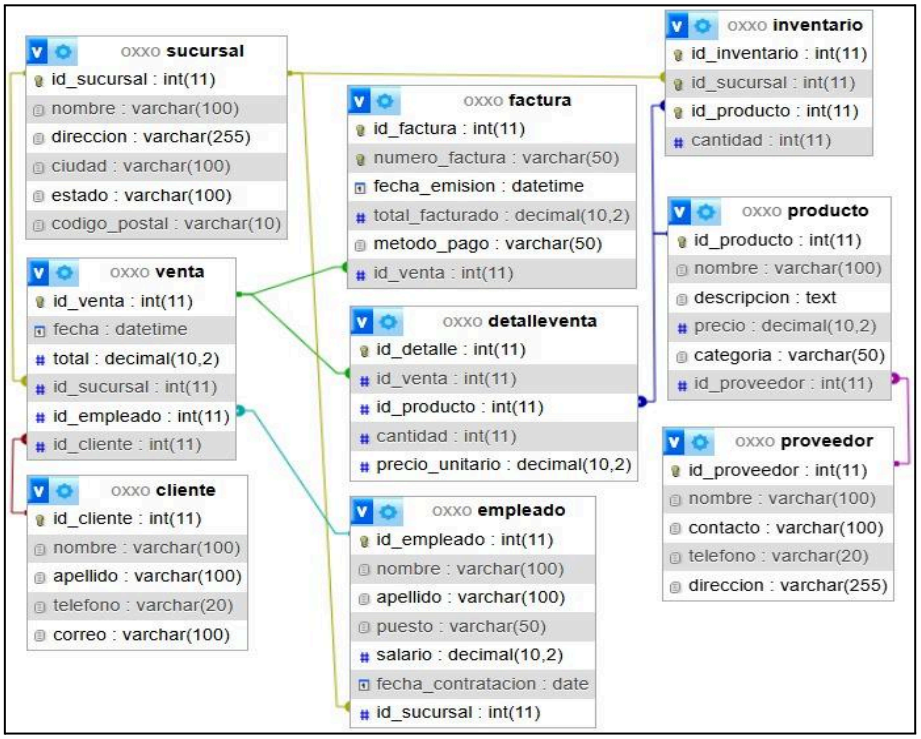


Fig 18. Imagen de los datos del servidor de producción de la instancia Ec2. Diseño propio.

6.1.2.1. Página web: Oxxo

Producto	Descripción	Cantidad	Precio unitario	Total de venta
M&M's 40g	Dulces de chocolate con cobertura	9	$25.00	$225.00
Cigarros Marlboro Rojo (20)	Cajetilla de cigarros	10	$50.25	$502.50
Sabritas Clasicas	Papas fritas	10	$12.00	$120.00
Agua Ciel 1L	Agua natural embotellada	2	$10.00	$20.00
Galletas Oreo	Galletas de chocolate con crema	7	$18.50	$129.50
Huevo blanco 12 piezas	Huevo fresco	6	$85.00	$510.00
Cheetos Torciditos 45g	Botana de maíz con queso	5	$16.00	$80.00
Chocolate Hershey's	Barra de chocolate 40g	6	$14.00	$84.00
Pepsi 355ml	Refresco en lata	10	$25.00	$250.00
M&M's 40g	Dulces de chocolate con cobertura	6	$25.00	$150.00
Papel Higiénico Regio (4 rollos)	Papel sanitario	10	$39.50	$395.00
Sabritas Clasicas	Papas fritas	1	$12.00	$12.00
Barritas Fresa Bimbo	Barritas rellenas de fresa	1	$24.00	$24.00
Red Bull 250ml	Bebida energética	8	$57.00	$456.00
Chocolate Hershey's	Barra de chocolate 40g	1	$14.00	$14.00
Cloralex 1L	Cloro multiusos	9	$43.00	$387.00
Papel Higiénico Regio (4 rollos)	Papel sanitario	1	$39.50	$39.50

Fig 19. Imagen de la página web con los datos de OXXO. Diseño propio.

En la figura 19, podemos ver la interfaz principal del cliente Oxxo, aquí podemos distinguir las ventas al público en general en tiempo real.

6.1.3. Servidor Windows Server 2022

Este servidor de producción simula las transacciones generadas en una tienda llamada “Foods”, las cuales incluyen la compra-venta de productos orgánicos al público en general. Una tienda de productos orgánicos es un negocio dedicado a la venta de productos certificados como orgánicos, es decir, que se cultivan y producen sin pesticidas, fertilizantes artificiales ni organismos genéticamente modificados, respetando los ciclos naturales y el medio ambiente.

Este servidor contiene una base de datos que simula la administración de la compra-venta de los diferentes productos dentro de la tienda Foods. Por ello cuenta con diferentes tablas, cada una de ellas con un propósito diferente. A su vez se desarrolló un programa que

simula las transacciones de compra de productos con los proveedores y venta de productos al público en general.

Base de datos
En esta base de datos se refleja la relación que existe entre los diferentes departamentos dentro de la tienda Foods. Un producto pertenece a una categoría y tiene un proveedor. Una venta la realiza un empleado a diferentes clientes y puede tener una factura. Cada venta contiene varios detalles de venta y una compra se hace a un proveedor y se detalla en detalle la compra.

En la figura 20, podemos distinguir cada una de las tablas que componen la base de datos de este cliente.
Las tablas con las que cuenta la base de datos son las siguientes:

- **“Sucursal”**
  Almacena la información de cada tienda Foods física para saber dónde se realizó una venta, dónde trabaja un empleado o dónde hay inventario.
  - **“id_sucursal”**: Identificador único de la sucursal.
  - **“nombre”**: Nombre asignado a la sucursal.
  - **“direccion”**
  - **“ciudad”**
  - **“estado”**
  - **“codigo_postal”**: Ubicación detallada.

- **“Empleado”**
  Guarda los datos de cada empleado que trabaja en una sucursal para registrar qué empleado hizo una venta y a que sucursal pertenece.
  - **“id_empleado”**: Identificador del empleado.
  - **“nombre”**: Datos personales.
  - **“apellido”**: Datos personales.
  - **“puesto”**: Cargo (ej. cajero, gerente).
  - **“salario”**: Sueldo mensual.
  - **“fecha_contratacion”**: Cuándo comenzó a trabajar.
  - **“id_sucursal”**: A qué sucursal está asignado.

- **“Proveedor”**
  Representa a las empresas que suministran productos a la tienda. Permite saber de dónde proviene un producto.
  - **“id_proveedor”**: ID del proveedor.
  - **“nombre”**
  - **“contacto”**
  - **“telefono”**
  - **“direccion”**: Datos de contacto.

- **“Producto”**
  Contiene la información de los artículos disponibles para la venta. Esta tabla está relacionada con inventario, ventas y proveedores.

- **“id_producto”**: ID del producto.
- **“nombre”**
- **“descripcion”**: Nombre y detalles del producto.
- **“precio”**: Precio de venta.
- **“categoria”**: Tipo de producto (bebidas, botanas, etc.).
- **“id_proveedor”**: Quién lo suministra.

- **“Inventario”**
  Controla la cantidad de productos por sucursal sirve para verificar la disponibilidad de productos en cada tienda.
  - **“id_inventario”**: ID único del registro.
  - **“id_sucursal”**: Dónde está el inventario.
  - **“id_producto”**: Qué producto.
  - **“cantidad”**: Cuánto stock hay disponible.

- **“Cliente”**
  Registra información opcional de clientes (para facturación o fidelización).
  - **“id_cliente”**: ID del cliente.
  - **“nombre”**
  - **“apellido”**
  - **“telefono”**
  - **“correo”**: Datos de contacto.

- **“Venta”**
  Representa cada compra realizada por un cliente en una sucursal. Esta tabla está relacionada con clientes, empleados, sucursales y detalles de venta.
  - **“id_venta”**: ID de la venta.
  - **“fecha”**: Cuándo se realizó.
  - **“total”**: Monto total.
  - **“id_empleado”**
  - **“id_cliente”**: Actores involucrados.

- **“DetalleVenta”**
  Registra los productos específicos incluidos en cada venta, nos sirve para saber qué se vendió, en qué cantidad y a qué precio.
  - **“id_detalle”**: ID del detalle.
  - **“id_venta”**: A qué venta pertenece.
  - **“id_producto”**: Qué producto se vendió.
  - **“cantidad”**: Cuántas unidades.
  - **“precio_unitario”**: Precio por unidad en esa venta.

- **“Factura”**
  Genera un comprobante fiscal de cada venta (para contabilidad o cliente). Es para asociar ventas con un documento fiscal.
  - **“id_factura”**: ID de la factura.
  - **“numero_factura”**: Código único de la factura.

- **“fecha_emision”**: Cuándo se generó.
- **“total_facturado”**: Monto facturado.
- **“metodo_pago”**: Forma de pago (efectivo, tarjeta, etc.).
- **“id_venta”**: Venta asociada.

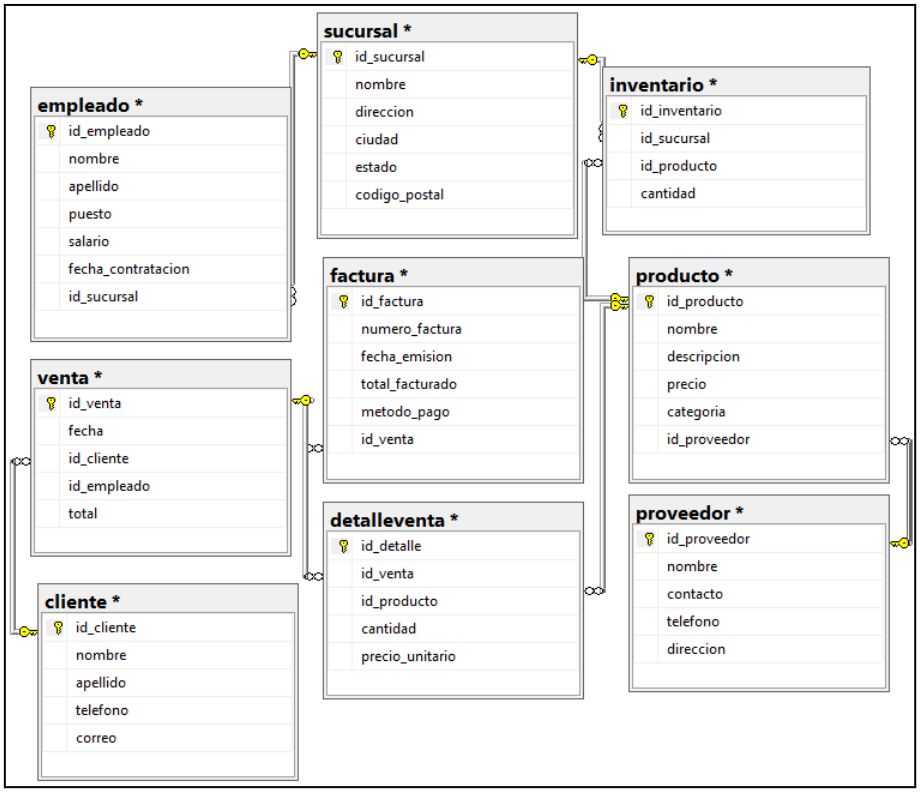


Fig 20. Imagen de los datos del servidor de producción de la instancia Ec2. Diseño propio.

6.1.3.1. Página web: Foods

Producto	Descripción	Cantidad	Precio unitario	Total de venta
Vinagre	Vinagre de Manzana Great Value Orgánico 500 ml	8	$62.50	$500.00
Pechuga sin hueso	Pechuga sin hueso orgánica Tru libre pastoreo 550 g	9	$149.50	$1345.50
Molida magra	Molida magra orgánica Tru libre pastoreo 350 g	4	$89.00	$356.00
Aguacate	Hass Orgánico 800g	6	$89.00	$534.00
Panqué Deorno	Panqué Deorno con chispas de chocolate 100 g	9	$26.50	$238.50
Jitomate	Jitomate uva orgánico Campo Vivo 280 g	4	$52.50	$210.00
Aceite de coco	Aceite de coco Aires de Campo orgánico extra virgen 473 ml	1	$168.00	$168.00
Tofu	Tofu Natural Orgánico 250g	4	$66.50	$266.00
Pechuga de Pollo	Pechuga de Pollo sin Hueso Orgánica por kg	7	$318.00	$2226.00
Tomate	Tomate uva Naturesweet Cherubs 467 g	4	$68.00	$272.00
Milanesa de pollo	Milanesa de pechuga de pollo orgánica 850 g	10	$324.00	$3240.00
Jugo de Naranja	Jugo de Naranja Marketside 1.89 L	2	$95.00	$190.00
Pechuga de Pollo	Pechuga de Pollo sin Hueso Orgánica por kg	9	$318.00	$2862.00
Arroz	Arroz Aires de Campo Orgánico 1 kg	1	$96.50	$96.50
Aguacate	Hass Orgánico 800g	1	$89.00	$89.00
Jugo de naranja	Jugo de naranja apio y nopal Frutos de Vida 320 ml	1	$24.90	$24.90

Fig 21. Imagen de la página web con los datos de Foods. Diseño propio.

En la figura 21, podemos ver la interfaz principal del cliente Foods, aquí podemos distinguir las ventas simuladas de cada uno de los productos que maneja.

6.2. Servidor Bastión

Este es el servidor central del sistema, el cual contiene, la base de datos con todos los detalles de los dueños de diferentes negocios, además de los servidores de producción asociados a cada uno de esos negocios. Cada uno de estos servidores contiene un sistema operativo igual o diferente al anterior y un servidor web que podría diferir también.

6.2.1. Servidor Windows Server 2019 Datacenter

Este es el servidor orquestador, este servidor es la pieza central del sistema, dado que contiene todos los detalles de las diferentes empresas, sus usuarios, sus servidores de producción y los detalles concernientes a cada uno de ellos.
También se configuró el entorno de Python y el ruteo del dominio orquestador.top al API REST del SBO.

6.2.1.1. Base de datos

En esta base de datos se refleja la relación que existe entre los diferentes departamentos dentro de una ferretería. Un producto pertenece a una categoría y tiene un proveedor. Una venta la realiza un cliente y puede tener una factura. Cada venta contiene varios detalles de venta y una compra se hace a un proveedor y se detalla en detalle la compra (figura 22).
Las tablas con las que cuenta la base de datos son las siguientes:

- **“Usuarios”**
  Registro de todos los tipos de usuarios que pueden acceder al sistema.
  - **“id”**: Identificador único.
  - **“nombre”**: Nombre completo del usuario.
  - **“username”**: Nombre de usuario para login.
  - **“password”**: Contraseña almacenada en formato hash por seguridad.
  - **“estatus”**: Estado del usuario (activo, inactivo, etc.).
  - **“userLevel”**: Este campo considera los siguientes tipos de usuarios:
    - 1: Usuario que únicamente puede entrar a ver el estatus de los servidores con los que se encuentre relacionado.
    - 2: Usuario que puede cambiar el estatus de los servidores de producción a los que tenga acceso. Este tipo de usuario está pensado para un tipo de usuario administrador, el cual será designado por las distintas empresas para que opere los diferentes servidores con los que cuenten.
    - 3: Super usuario, el cual puede registrar, a otros usuarios, negocios, servidores, así como registrar todos los detalles relacionados con los mismos. Además, puede relacionar los servidores previamente registrados con los usuarios que tendrán acceso a ellos.

- **“Negocios”**
  Representa las sucursales o negocios que pertenecen a usuarios.
  - **“id”**: Identificador único.
  - **“nombre_sucursal”**: Nombre de la sucursal o negocio.
  - **“usuario_id”**: Dependiendo del nivel de usuario que se tenga registrado, éstos podrán únicamente ver el estatus de los servidores o poder operarlos para cambiar el estatus de los mismos.

- **“Negocios_servidores”**
  Tabla que relaciona los negocios registrados con los servidores asociados a los mismos, por lo cual relaciona un tipo de servidor registrado con otro negocio previamente registrado.
    - **“id”**: Identificador único.
    - **“negocio_id”**: Id único de un negocio previamente registrado.
    - **“servidor_id”**: Id único de un servidor previamente registrado.

- **“Servidores”**
  Contiene la información concerniente a los servidores de producción vinculados a los negocios de los diferentes clientes.
    - **“id”**: Identificador único.
    - **“nombre_servidor”**: Nombre o etiqueta del servidor de producción que se operará.
    - **“sistema_operativo”**: Sistema operativo instalado en el servidor seleccionado. Dentro de la presente propuesta se están considerando los sistemas operativos de Windows y Linux en alguna de sus variantes para que operen los servidores de producción.
    - **“base_datos”**: Información sobre el tipo de base de datos instalada.
    - **“pagina_web”**: URL de la página web alojada dentro de cada uno de los servidores de producción, misma que nos sirve para visualizar los balances de los productos que se están manejando dentro de cada una de ellas. Como se mencionaba antes, dentro de esta tesis se están considerando dos negocios, una es una ferretería y el segundo negocio es una tienda de conveniencia del tipo “oxxo” la cual realiza ventas al público en general.
    - **“procesador”**: Contiene el tipo de procesador que están utilizando los servidores de producción.
    - **“memoria_ram”**: Contiene la información relacionada al uso de la memoria ram de acuerdo a la carga de trabajo dentro de los servidores.
    - **“almacenamiento”**: Contiene la cantidad de almacenamiento disponible.
    - **“ip_direccion”**: Dirección IP del servidor.
    - **“estado”**: Estado actual del servidor (activo, inactivo, mantenimiento, etc.).
    - **“key_id”**: Es un código de acceso generado dentro de la plataforma de AWS el cual es una llave única por cada uno de los servidores.
    - **“access_key”**: Es una clave generada dentro de la plataforma de AWS la cual es una clave secreta de 68 dígitos, esta única a cada servidor.
    - **“region”**: Ubicación geográfica donde se encuentra registrado el servidor con el que se está trabajando. Actualmente estamos trabajando con dos regiones, las cuales son: “us-east-1” y “us-east-2”.
    - **“instanceId”**: Identificador único a cada una de las instancias creadas y utilizadas como servidores de producción.
    - **“accion_id”**: Este campo nos indica la última acción seleccionada y que se realizó en un servidor de producción determinado. Dichas acciones se escogen al seleccionar uno de los botones adjuntos a los diferentes servidores registrados dentro de la plataforma. Al pulsar alguno de estos botones, se toma la información de la base de datos registrada del servidor, y

se adjunta con la acción que se desea realizar, se manda a una consola que recibe toda la información como parámetros y ésta ejecuta el comando seleccionado.

- ON: Este comando encenderá el servidor de producción seleccionado.
- OFF: Este comando realizará el apagado del servidor seleccionado.

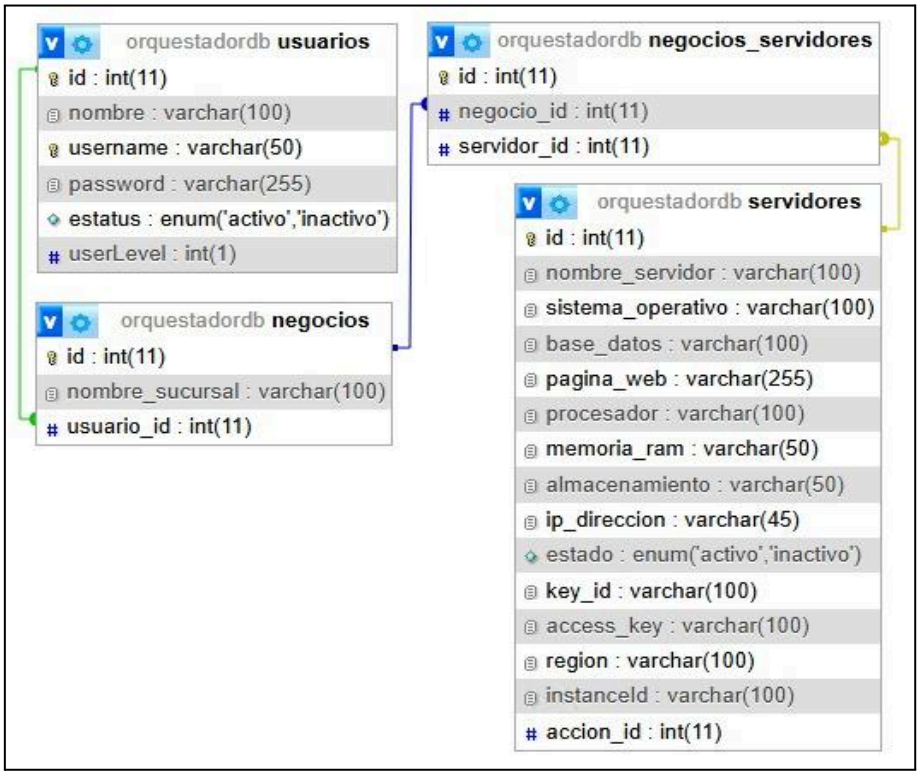
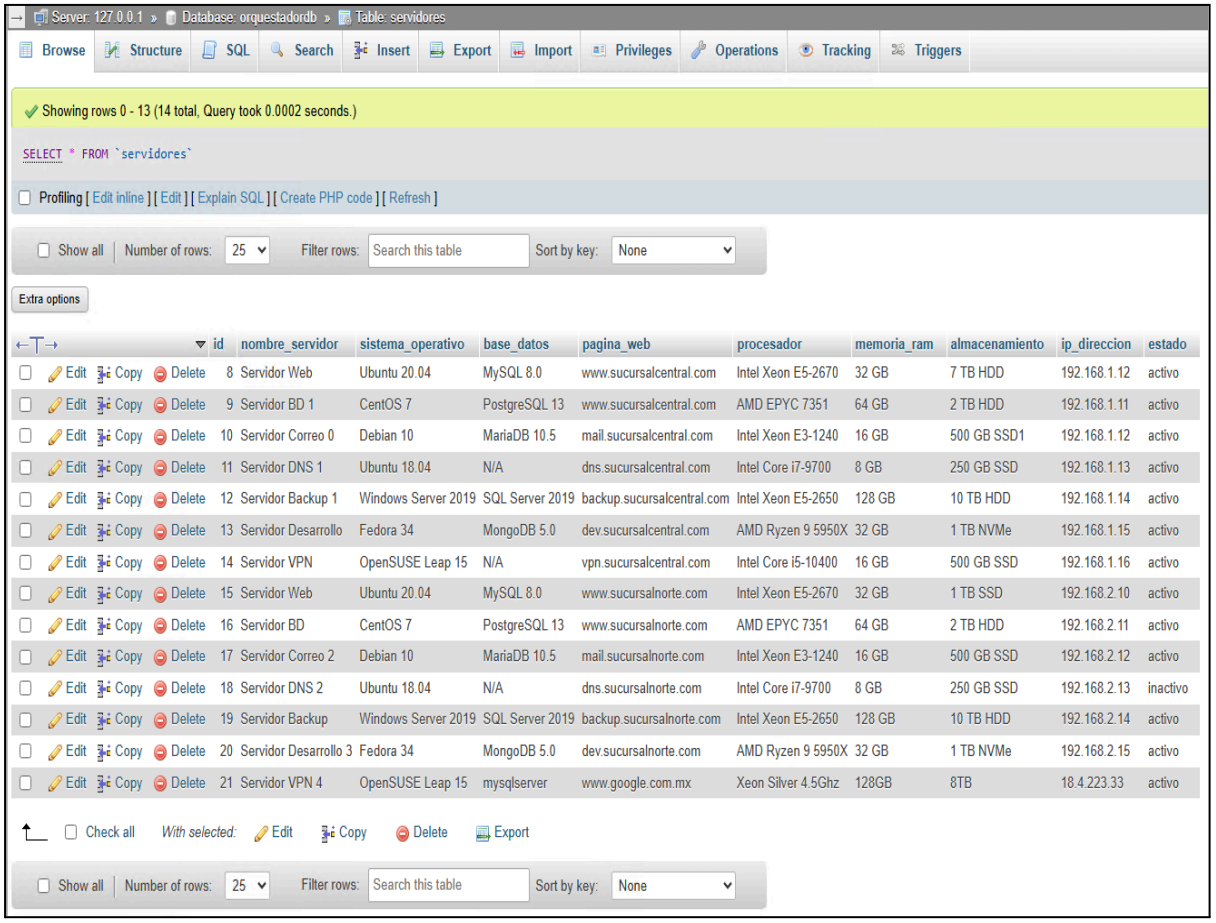


Fig 22. Imagen de la base de datos del orquestador. Diseño propio.

En la Figura 19, podemos ver el diagrama general del sistema que se propone, dicho diagrama está compuesto por diferentes partes las cuales se describieron anteriormente.

Vista general de las tablas

En la siguiente imagen podemos apreciar como se ve la vista general de la tabla servidores, misma que contiene toda la información de los diferentes servidores de producción de los negocios que se tengan registrados. Cabe destacar que los servidores que actualmente se muestran son ficticios en su mayoría, sin embargo, 2 de ellos son reales, mismo que son en los que se desplegaron los negocios de ferretería y tienda de conveniencia para mostrar el funcionamiento de la plataforma. En la figura 23, tenemos un vistazo general de la tabla **“servidores”** la cual muestra cada una de las características asociadas a cada uno de los servidores listados de los clientes.



Server: 127.0.0.1 » Database: orquestadordb » Table: servidores

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Tracking | Triggers

Showing rows 0 - 13 (14 total, Query took 0.0002 seconds.)

```
SELECT * FROM `servidores`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

	id	nombre_servidor	sistema_operativo	base_datos	pagina_web	procesador	memoria_ram	almacenamiento	ip_direccion	estado
Edit Copy Delete	8	Servidor Web	Ubuntu 20.04	MySQL 8.0	www.sucursalcentral.com	Intel Xeon E5-2670	32 GB	7 TB HDD	192.168.1.12	activo
Edit Copy Delete	9	Servidor BD 1	CentOS 7	PostgreSQL 13	www.sucursalcentral.com	AMD EPYC 7351	64 GB	2 TB HDD	192.168.1.11	activo
Edit Copy Delete	10	Servidor Correo 0	Debian 10	MariaDB 10.5	mail.sucursalcentral.com	Intel Xeon E3-1240	16 GB	500 GB SSD1	192.168.1.12	activo
Edit Copy Delete	11	Servidor DNS 1	Ubuntu 18.04	N/A	dns.sucursalcentral.com	Intel Core i7-9700	8 GB	250 GB SSD	192.168.1.13	activo
Edit Copy Delete	12	Servidor Backup 1	Windows Server 2019	SQL Server 2019	backup.sucursalcentral.com	Intel Xeon E5-2650	128 GB	10 TB HDD	192.168.1.14	activo
Edit Copy Delete	13	Servidor Desarrollo	Fedora 34	MongoDB 5.0	dev.sucursalcentral.com	AMD Ryzen 9 5950X	32 GB	1 TB NVMe	192.168.1.15	activo
Edit Copy Delete	14	Servidor VPN	OpenSUSE Leap 15	N/A	vpn.sucursalcentral.com	Intel Core i5-10400	16 GB	500 GB SSD	192.168.1.16	activo
Edit Copy Delete	15	Servidor Web	Ubuntu 20.04	MySQL 8.0	www.sucursalnorte.com	Intel Xeon E5-2670	32 GB	1 TB SSD	192.168.2.10	activo
Edit Copy Delete	16	Servidor BD	CentOS 7	PostgreSQL 13	www.sucursalnorte.com	AMD EPYC 7351	64 GB	2 TB HDD	192.168.2.11	activo
Edit Copy Delete	17	Servidor Correo 2	Debian 10	MariaDB 10.5	mail.sucursalnorte.com	Intel Xeon E3-1240	16 GB	500 GB SSD	192.168.2.12	activo
Edit Copy Delete	18	Servidor DNS 2	Ubuntu 18.04	N/A	dns.sucursalnorte.com	Intel Core i7-9700	8 GB	250 GB SSD	192.168.2.13	inactivo
Edit Copy Delete	19	Servidor Backup	Windows Server 2019	SQL Server 2019	backup.sucursalnorte.com	Intel Xeon E5-2650	128 GB	10 TB HDD	192.168.2.14	activo
Edit Copy Delete	20	Servidor Desarrollo 3	Fedora 34	MongoDB 5.0	dev.sucursalnorte.com	AMD Ryzen 9 5950X	32 GB	1 TB NVMe	192.168.2.15	activo
Edit Copy Delete	21	Servidor VPN 4	OpenSUSE Leap 15	mysqlserver	www.google.com.mx	Xeon Silver 4.5Ghz	128GB	8TB	18.4.223.33	activo

Check all | With selected: Edit | Copy | Delete | Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Fig 23. Imagen a la vista general de la tabla de servidores. Diseño propio.

6.2.1.2. Plataforma web

En esta sección podemos encontrar cada una de las partes que componen a la plataforma web del Servidor bastión. Cada una de ellas se encuentra descrita a continuación.

6.2.1.2.1. Certificados SSL

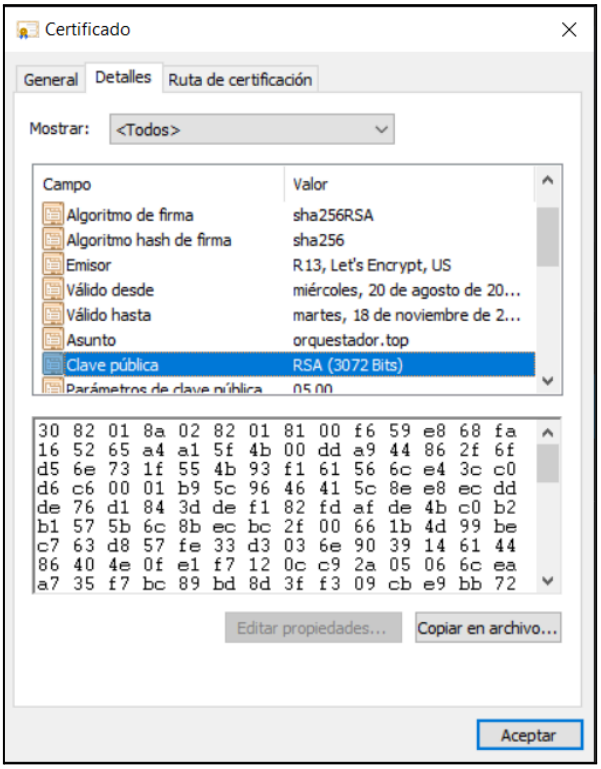


Fig 24. Imagen del certificado instalado en el Servidor Bastión para el establecimiento de comunicación con los clientes.

En la figura 24, tenemos un vistazo del certificado instalado en el servidor para llevar a cabo la comunicación entre los clientes y el de una forma segura. Algunas de las características del certificado SSL son las siguientes:

- Emitido para: **orquestador.top**
  Este es el nombre de dominio DNS donde se encuentra alojada la página principal del sistema.
- Emitido por: **R13**
  Esta es la autoridad certificadora.
- Válido desde: **20/08/2025**
- hasta: **18/11/2025**

6.2.1.2.2. Sistema de autenticación con JWT

Un sistema JWT (JSON Web Token) es una arquitectura de autenticación y autorización basada en tokens que permite a los servidores emitir objetos JSON firmados que los clientes pueden usar para demostrar su identidad. Este es un estándar abierto (RFC 7519) que define un método compacto, autocontenido y seguro para transmitir información entre dos partes como un objeto JSON. El token consta de tres secciones codificadas en Base64URL, separadas por puntos. Es un sistema ampliamente utilizado en aplicaciones web modernas para mantener sesiones de usuario sin necesidad de almacenar información en el servidor (sesiones sin estado o stateless). Este es un sistema de autenticación robusto que proporciona seguridad y escalabilidad en nuestro sistema. El sistema sigue un flujo estándar de autenticación y sus componentes los que se listan a continuación.
También se implementó el esquema de autenticación JWT así como las rutinas de gestión de usuarios (exclusivas del Nivel 2). Se desarrolló la lógica de RBAC para que la API REST consulte los derechos del usuario (nivel_acceso) y garantice la cobertura de funcionalidad del 100% de los comandos solo para usuarios Nivel 1.

Autenticación

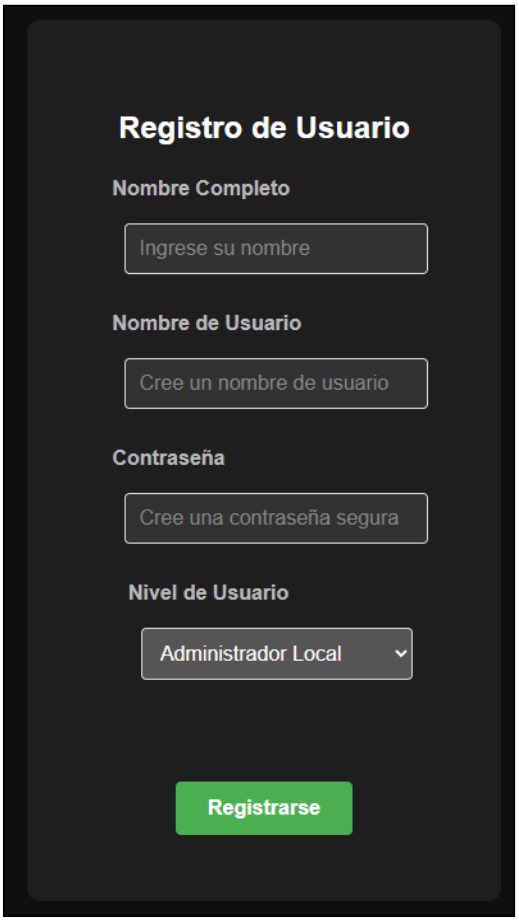


Fig 25. Registro: Pantalla para el registro de usuarios nuevos. Diseño propio.

- Ruta al registro del sistema: https://orquestador.top:3000/register
- Registro: En la figura 25 se puede apreciar cómo se lleva a cabo el registro de un usuario dentro de la plataforma. El registro de nuevos usuarios se debe hacer con un super usuario (de nivel 2), dado que este es el único que puede crear las relaciones entre usuarios y los servidores a los que tendrá acceso. Esto pone un nivel de seguridad más al sistema, dado que no cualquier persona se puede registrar en la plataforma, solo los usuarios que estén pre aprobados por un usuario administrador de rango mayor.
- Inicio de sesión: En la siguiente imagen 26, se puede apreciar el inicio de sesión para los usuarios registrados en la plataforma, éste es general a todos los tipos de usuario.

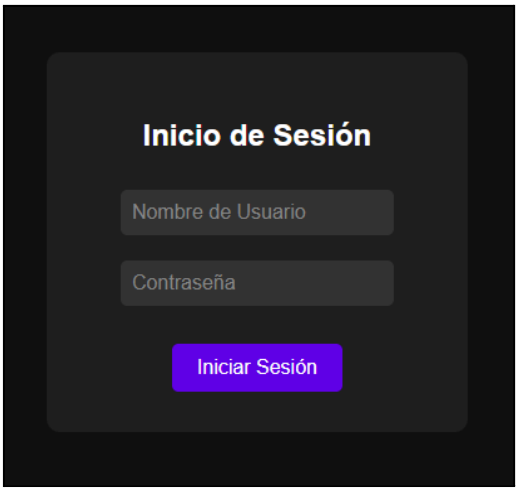


Fig 26. Login: El usuario inicia sesión con sus credenciales. Diseño propio.

- Verificación: El servidor valida las credenciales en la base de datos, mismas que todo el tiempo se encuentran codificadas desde el momento de su inserción por medio de la función bcrypt.hash.
- Generación de Token: Si las credenciales introducidas por el usuario son válidas, se genera un JWT con una validez limitada por un rango de tiempo y firmado con la clave privada del servidor.
- Envío de Token: El token se envía al cliente.
- Acceso Protegido: El cliente incluye el token en solicitudes posteriores.
- Verificación de Token: El servidor verifica el token en cada solicitud protegida.

Fig 26a. Pantalla para la administración de los Servidores de un cliente final. Diseño propio.

En la figura 26a, podemos apreciar la vista general de los servidores de un cliente, sin embargo en la figura 26b, podemos ver la validación de tokens que se está realizando en tiempo real, y si alguno llegara a cambiar la sesión del cliente se cierra.

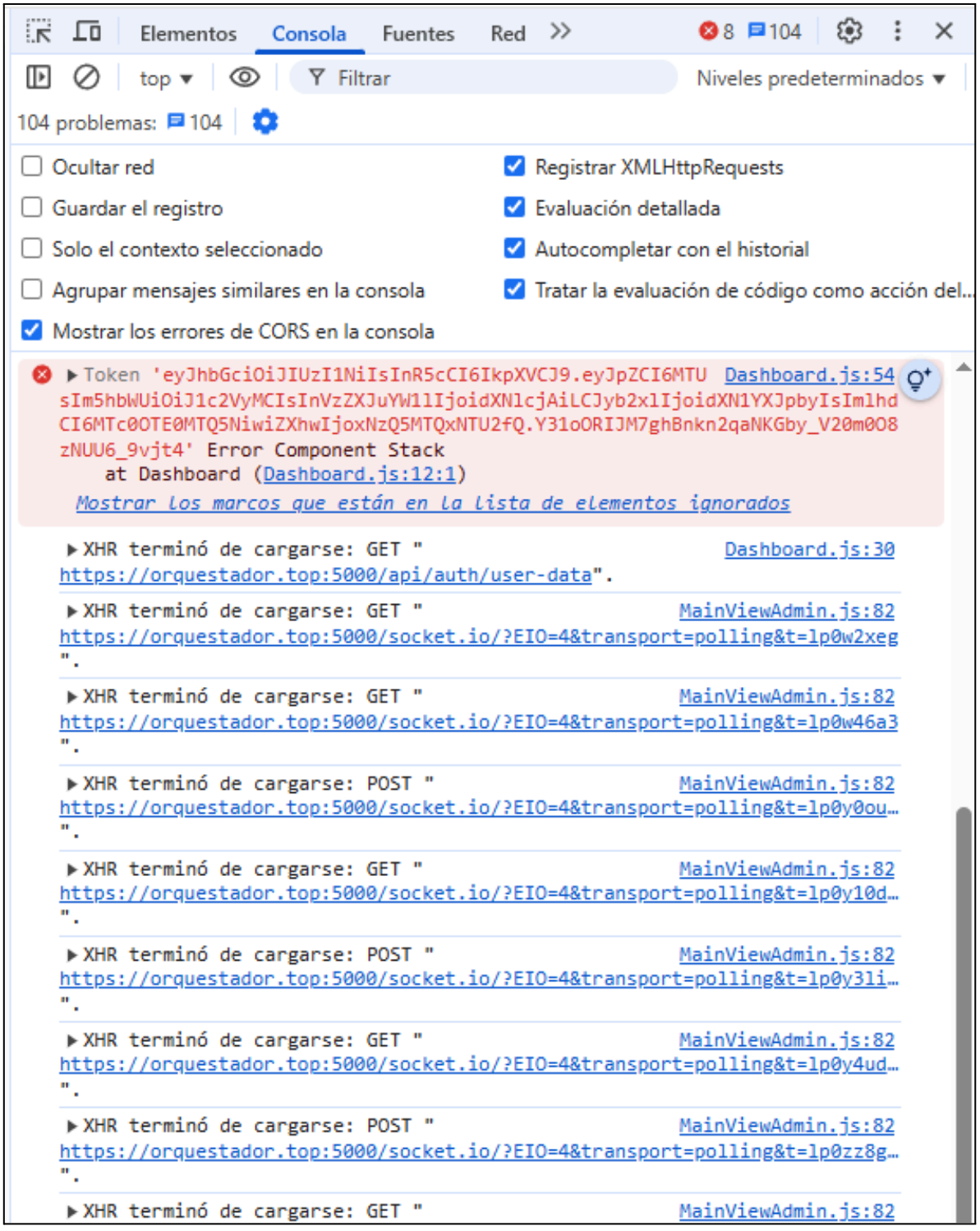


Fig 26b. Imagen de la depuración con el Token generado. Diseño propio.

Estructura del Token JWT

Cada uno de los tokens se componen de tres partes:

- plaintext
- Header: Especifica el algoritmo de encriptación (ej: HS256)
- Payload: Contiene los "claims" (datos del usuario y metadatos)f
- Signature: Firma digital que verifica la autenticidad del token
- Ejemplo: Token generado por el servidor:

  “eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MTUsIm5hbWUiOiJ1c2VyMCIsInVzZXJuYW1lIjoidXNlcjAiLCJyb2xlIjoidXN1YXJpbyIsImlhdCI6MTc0OTA4MTI4MiwiZXhwIjoxNzQ5MDgxMzQyfQ.Wkf3T6-qLXzaycXgYyqnhownEGkCQqQyjFX1fsTV9Z8”

Este sistema de autenticación basado en JWT proporciona una base sólida y segura para la aplicación, permitiendo un control de acceso granular y manteniendo un buen rendimiento incluso con un gran número de usuarios.

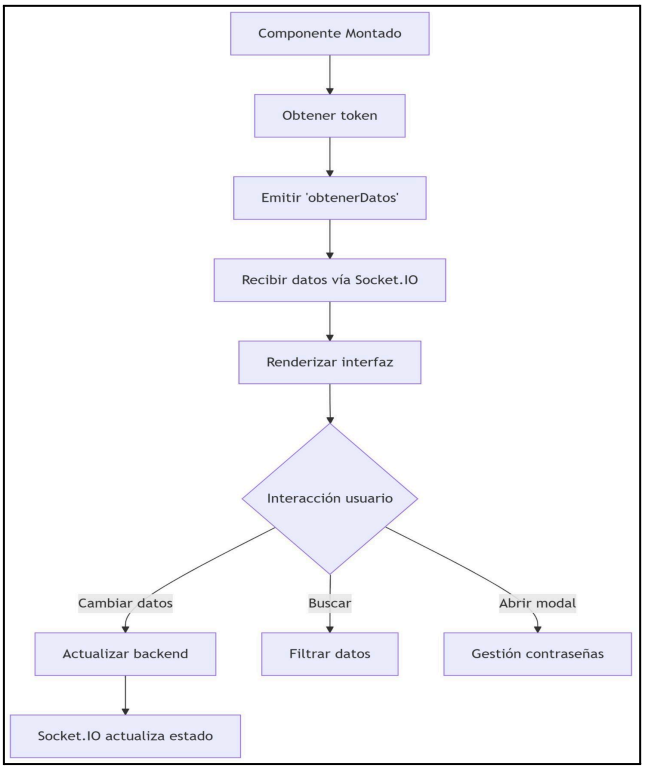


Fig 27. Diagrama de la estructura del token. Diseño propio.

En la primera parte del diagrama, se muestra la autenticación la cual consta de dos pasos, con lo cual se propone invalidar el acceso no autorizado a la plataforma a personal no registrado en la misma. Además de la autenticación de dos pasos, se propone utilizar conexiones seguras, es decir utilizar protocolos https, para que la conexión vaya cifrada entre el administrador de la plataforma y la misma.
En la figura 27, podemos distinguir la estructura para la gestión de los tokens que se manejan con cada uno de los clientes de la plataforma.

6.2.1.3. Sitio web

En la imagen 28, se puede apreciar la vista general de los clientes (negocios) registrados dentro del sistema junto a los usuarios que los pueden operar. En la misma podemos apreciar una lista de 7 negocios con una locación hipotética de cada uno de ellos. Para el caso mostrado, el ‘user0’ tiene los derechos de operación sobre todos los servidores asociados a estos negocios.

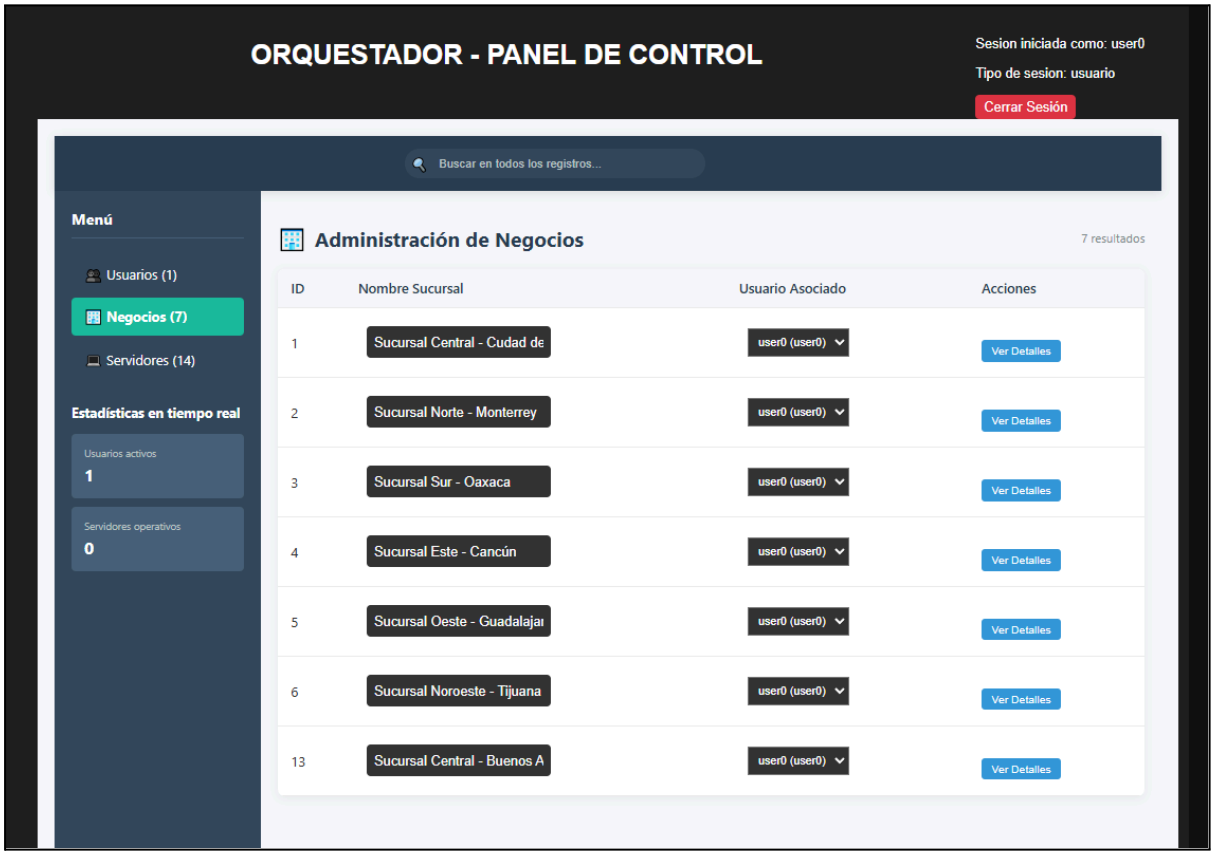


Fig 28. Diseño con el listado de Servidores de un cliente final. Diseño propio.

6.2.1.4. Clientes

En la siguiente imagen (fig 29), podemos apreciar la lista de los usuarios actualmente registrados en la plataforma, en este podemos ver que los usuarios son genéricos ‘user0’, ‘user1’, ‘user2’, esto tiene una razón de ser, el número asociado al nickname de el usuario nos muestra el nivel de usuario. Como se había mencionado con anterioridad, el nivel de usuario 0, únicamente tiene acceso a ver el estatus de los servidores a los que tenga derecho, pero no puede operarlos, únicamente puede ver como se encuentran trabajando.
Por otra parte el ‘user1’ tiene acceso a otros servidores pero este tipo de usuario si puede gestionarlos, es decir, puede cambiar el estatus de estos, puede prender y apagar estos servidores, por lo cual sus derechos son más elevados que los del usuario 0.
A su vez podemos observar que existe un ‘user2’ este es un super usuario, éste es el único usuario que puede registrar a otros usuarios así como los servidores a los que tendrá acceso. También puede agregar más usuarios y servidores generando más relaciones entre ellos o bloquear las existentes, dar de baja usuarios y actualizar la información de los servidores existentes.

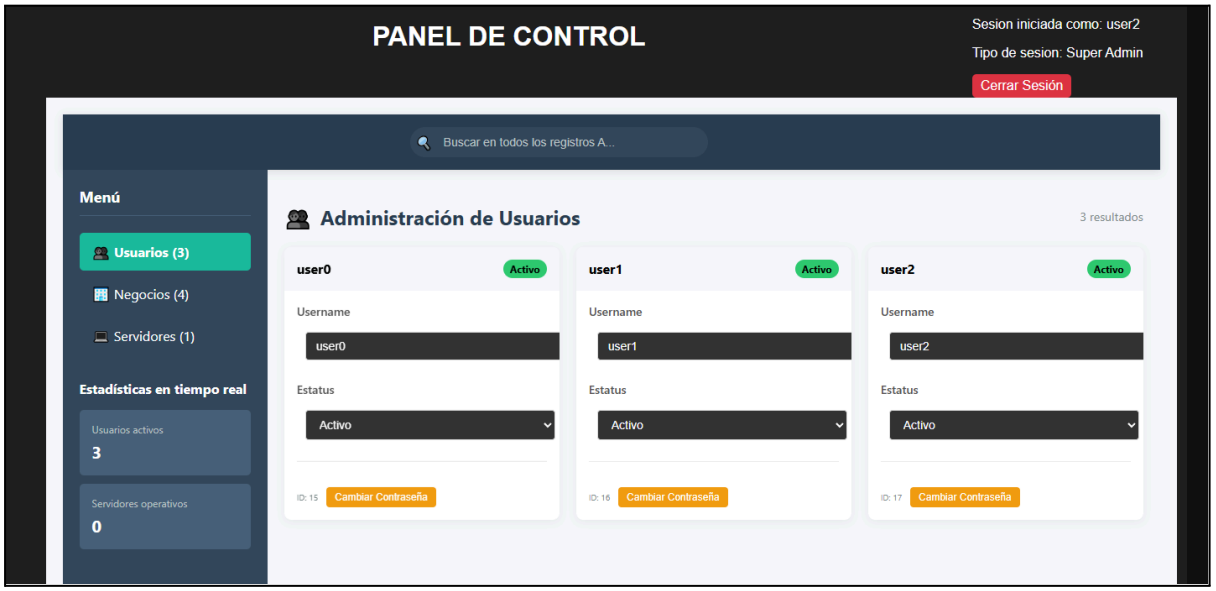


Fig 29. Pantalla para la administración de usuarios nivel 1 y 2. Diseño propio.

AWS

Como se mencionaba anteriormente, la nube pública en la que se tienen registrados todos los servidores es la nube de Amazon y el tipo de instancias que se escogieron para realizar el despliegue de los servidores son **"EC2"** en la imagen que se muestra a continuación, se pueden apreciar todos los detalles concernientes al servidor central Bastión, sobre el que se está desplegando toda la la plataforma.

6.2.1.5. Monitoreo en tiempo real

El servidor bastión, mismo que contener la lista de clientes hipotéticos registrados dentro de la plataforma, se encontrará escaneando de forma periódica cada uno de los servidores que contenga registrados dentro de sí, esto con la finalidad de en todo momento contener información fidedigna sobre el estado de cada uno de ellos y poder realizar la operación del cambio de estatus en cada uno de los servicios asociados a los mismos.

En cuanto el administrador se conecte al servidor, este podrá revisar el último estatus asociado de cada uno de los servidores, dado que este se va actualizando de forma periódica. A su vez el administrador podrá realizar el cambio de estatus de un servidor toda vez que reciba la instrucción del cliente final para realizar dicho cambio, con lo cual de forma intuitiva y transparente deberá mandar los comando asociados al cambio de estatus de un servidor en específico de un cliente y negocio en específico pudiendo saber en todo momento el estatus de cada uno de los servicios asociados a la infraestructura de SAP en todo momento.

A su vez se desarrolló la codificación de las rutinas Python que implementa la lógica del Diagrama de Secuencia UML para los comandos Start/Stop a través de la conexión remota.

También se desarrolló el programa monitor de los servidores remotos de los clientes el cual fue construido en Python y revisa de forma continua el estatus de cada servidor de producción. Cabe destacar que el script debe consultar la DB SQL para obtener las credenciales y el estado anterior, conectarse al servidor de producción, y actualizar el estado en la DB solo si se detecta un cambio, asegurando la Latencia de Actualización <5 segundos para la interfaz. Para esta parte de la implementación, es imperativo definir los roles con jerarquía y permisos claros (modelo RBAC - Role-Based Access Control), super

Usuario (Nivel 2 - Administrador de la Plataforma). Este tipo de usuario, tiene los permisos de creación, modificación y eliminación de cuentas de usuario, negocios y registros de servidores. Su propósito es el mantenimiento del repositorio de metadata y la infraestructura del Servidor Bastión.
Para el usuario de nivel 1, este es un gestor de Operaciones (usuario administrativo), este tipo de usuario puede ejecutar los comandos de Encendido/Apagado de los servidores de producción asignados a su negocio. Su propósito es la optimización directa de costos operativos por indicación del cliente final al encender o apagar los servidores de producción del mismo.
EL usuario monitor (Nivel 0 - Usuario de Lectura), este usuario únicamente puede visualizar el estatus en tiempo real (Encendido, Apagado, Ejecutando) y pedir que un usuario de nivel administrador opere alguno de sus servidores si lo requiere. Su propósito es el seguimiento y auditoría del funcionamiento del orquestador.

6.2.1.6. Arquitectura de los clientes

Las arquitecturas utilizadas por cada uno de los clientes registrados dentro de la plataforma web, podrá variar, entre cada uno de ellos e inclusive entre cada uno de los negocios asociados a un cliente en específico, dichas arquitecturas se describen más adelante en el apartado “arquitectura de un servidor de producción” (figura 30).
Dicho lo anterior, la cantidad de negocios asociados a un cliente y la cantidad de servidores asociados a dichos negocios puede ser variable entre cada uno de ellos, pudiendo ser diferentes los sistemas operativos, gestores de bases de datos y más. Aunque en la presente propuesta solo se contempla la operación sobre servidores registrados sobre la nube de AWS, los sistemas operativos Windows Server, Linux y los gestores de bases de datos: Microsoft Sql Server y Oracle.

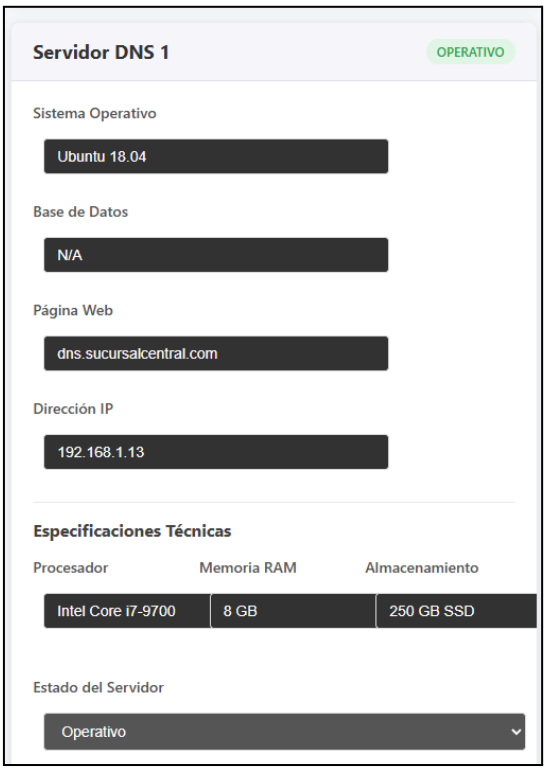


Fig 30. Pantalla de administración de un Servidor de producción. Diseño propio.

6.2.1.7. Servidores de Producción

Dependiendo del cambio de estatus que se quiera llevar a cabo sobre un servidor este debe seguir en general ciertos pasos para realizar su cambio de estatus (encender/apagar). Una vez que un determinado servidor reciba la orden del cambio de estatus, este debe comenzar a ejecutarla previa revisión de la autenticación asociada a cada uno de ellos.

6.2.1.7.1. Encendido

El encendido de un servidor conlleva la inicialización de una subrutina implementada en cada uno de los servidores previamente, la cual consta de que se autoinicia de forma automática una vez que cada servidor sea iniciado, con lo cual dicha subrutina, debe reportar el estatus del servidor al servidor Maestro (Bastión), acto seguido, una vez que el sistema operativo se encuentre completamente operativo, la subrutina debe desencadenar el encendido de el gestor de base de datos asociado al sistema SAP, toda vez que el estatus del SO lo permita y se encuentre en las condiciones óptimas para operar (figura 31). Concluida la fase anterior, se debe encender el servidor web que cada uno de los servidores tenga asociado, con lo cual una vez terminada dicha acción, la operatividad del sistema SAP asociada a dichos servidores quedará operativa y a la espera de las cargas que cada uno de los clientes tenga destinada hacia dichos servidores.

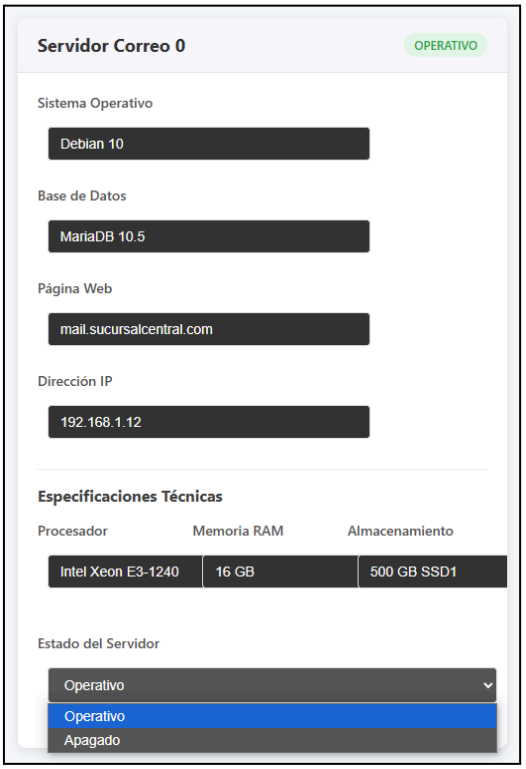


Fig 31. Pantalla de encendido de un Servidor de producción. Diseño propio.

6.2.1.7.2. Apagado

El proceso de apagado de cada uno de los servidores se realizará en el sentido inverso del anterior paso, con la diferencia de que cada uno de los comando asociados a cada una de las acciones anteriormente descritas se ejecutará, cuando la cargas asociadas a cada servidor lo permita, es decir, cada uno de los pasos se tratará de ejecutar dependiendo de la carga que cada servidor tenga en el momento de la recepción del comando de apagado. Dando prioridad al detenido pasivo de cada uno de los servicios y así evitar, que la información contenida en los servidores sufra algún tipo de corrupción, o pérdida de datos (figura 32).

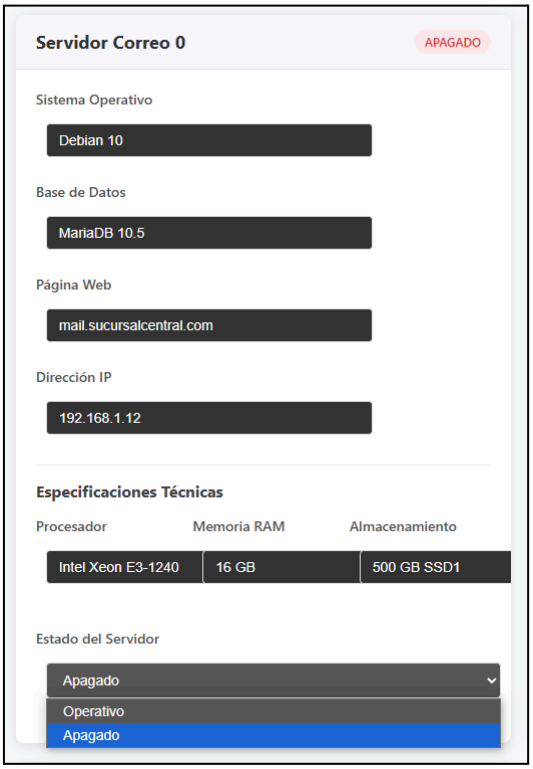


Fig 32. Pantalla de apagado de un Servidor de producción. Diseño propio.

6.2.1.7.3. Orquestador

La figura del "orquestador", dependiendo del contexto en el que nos encontremos trabajando, puede cambiar su significado. Dentro de la gestión de proyectos, se refiere a un gerente de proyecto que supervisa el avance general de cada una de las partes que lo componen. También puede ser un gestor que coordina procesos y recursos dentro de una organización para garantizar que todas las partes trabajen juntas de manera efectiva para alcanzar diferentes objetivos.

1) En el contexto tecnológico nos encontramos con una herramienta como “Kubernetes” la cual nos ayuda en la gestión de contenedores dado que administra contenedores y microservicios, actúa como orquestador al coordinar el despliegue y escalado de aplicaciones distribuidas [10].

Por lo cual en la presente propuesta nos referimos al “orquestador” como una herramienta de software que gestionará servidores y servicios así como aplicaciones para semi-automatizar el encendido y apagado de servidores en la nube.

Como en el apartado anterior se describió, el presente trabajo pretende gestionar y orquestar el funcionamiento de cada uno de los servidores registrados dentro de la plataforma, toda vez que las claves y permisos asociados a cada uno de ellos así lo permitan. El registro de cada uno de los clientes, arquitecturas, servidores así como las claves asociadas a cada servidor se debe llevar de forma meticulosa dado que un cambio en alguno de los campos derivará en la no operatividad de algún servidor según lo descrito (figura 33).

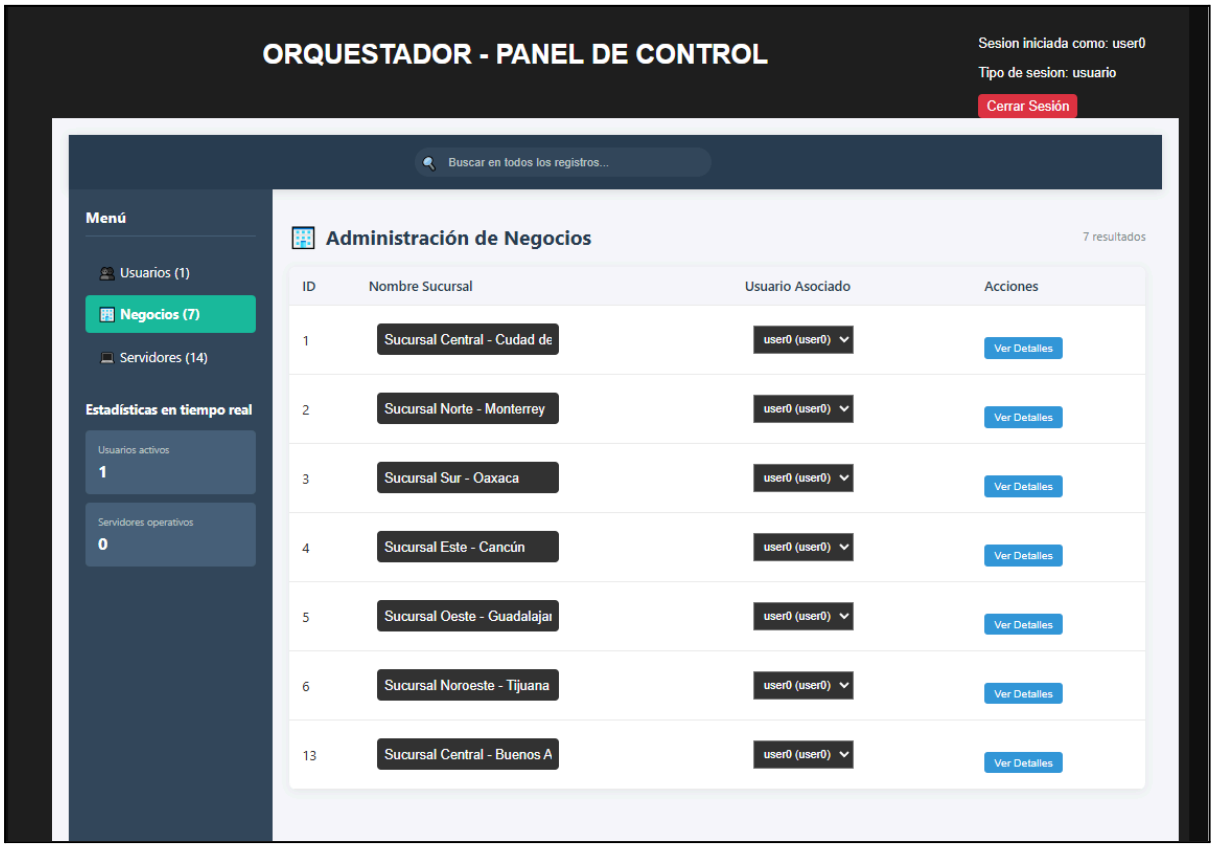


Fig 33. Pantalla de los negocios asociados a un usuario final. Diseño propio.

7. Resultados

Los resultados obtenidos por cada uno de los casos de estudios se muestran en la siguiente tabla comparativa.

Para el cálculo de la parte financiera, se documentó el costo base operativo semanal (CB) (24/7) para los Casos de Estudio. Se calculó el costo optimizado semanal (CO), aplicando los horarios de operación definidos por el SBO.

Parámetro	Caso de Estudio 1: Dev/QAS	Caso de Estudio 2: Corporativo Mediano QAS	Caso de Estudio 3: Corporativo Grande Dev
Instancias EC2 a Orquestar	3 (SO, DB, App)	5 (SO, 2DB, 2App)	8 (SO, 3DB, 4App)
Horas Requeridas de Ejecución (Semanal)	40 horas (8 horas/día x 5 días)	60 horas (12 horas/día x 5 días)	168 horas (24 horas/día x 7 días)
Horas Total de la Semana	168 horas	168 horas	168 horas
Horas de Subutilización (Ahorro Potencial)	**128 horas**	**108 horas**	**0 horas (Control)**
Costo por Hora de Instancia Promedio	$0.50 USD/hr	$1.20 USD/hr	$1.80 USD/hr
Costo Base (CB) Semanal (24/7)	0.50×168×3=$252.00	1.20×168×5=$1,008.00	1.80×168×8=$2,419.20
Costo Optimizado (CO) Semanal (Con SBO)	0.50×40×3=$60.00	1.20×60×5=$360.00	1.80×168×8=$2,419.20
Ahorro Absoluto Semanal	$192.00	$648.00	$0.00
Porcentaje de Ahorro (%)	76.19%	64.29%	0%

En el presente reporte podemos observar que se formaliza el cálculo del ahorro de costos, validando la hipótesis central de la tesis. Los casos de estudio 1 y 2 superan el umbral de ahorro del 35%, validando la parte económica de la hipótesis. La cual propuso una reducción de los costos asociados >35% en la renta asociada a los servidores de producción de los clientes registrados en la plataforma.

Para la validación de la calidad del servicio Qos, a continuación se muestra el formato con los resultados obtenidos.

Métrica de QoS	Caso de Estudio 1	Caso de Estudio 2	Caso de Estudio 3
Número Total de Ciclos Probados	200	250	**N/A (Prueba de Estrés)**
TPE Ciclo START (Promedio)	28.5 segundos	29.8 segundos	< 30 segundos
TPE Ciclo STOP (Promedio)	15.2 segundos	17.5 segundos	< 30 segundos
Número de Fallos Registrados	1 (Fallo de red esporádico)	2 (Error en API de AWS)	**N/A**
Tasa de Fallos en el Ciclo de Vida (TFC)	1/200 = 0,005%	2/250 = 0,008 %	< 1%
Resultado de la Validación (QoS)	**Éxito (Cumple)**	**Éxito (Cumple)**	---

Las pruebas de rendimiento y confiabilidad confirman que el Servidor Bastión Orquestador opera con un TPE ≤ 30$ segundos y una TFC ≤ 1%, validando la confiabilidad. En este reporte podemos apreciar que la eficiencia no compromete la confiabilidad del sistema.

Se hace hincapié en que dada la información sensible que se almacena y procesa en los servidores de clientes reales, por el momento no se puede aplicar la presente propuesta con los servidores mencionados. Es por esto que se define como un simulador de orquestación de servidores y servicios dado que nuestros datos son ficticios, sin embargo, el funcionamiento del sistema se está diseñando lo más apegado a lo real posible.

- Plataforma web montada en la nube AWS con una base de datos de clientes hipotéticos y las infraestructuras de servidores de producción asociadas a cada una de sus empresas.
- La plataforma debe ser capaz de administrar de forma discriminada cada uno de los servidores de producción de algún cliente dado.
- El sistema debe ser capaz de administrar cada servidor con la menor interacción humana posible.
- El sistema debe evitar el fallo en el inicio o apagado de cada servicio asociado a un determinado servidor.

8. Conclusión

Con el análisis de los resultados obtenidos en el presente trabajo de tesis, podemos concluir que ha logrado su objetivo general al diseñar, implementar y validar con éxito el Servidor Bastión Orquestador (SBO). Con el modelado UML se garantizó la correcta estructuración del software y su despliegue sobre la plataforma Windows Server 2019/SQL asegurando el acceso mediante el RBAC multinivel.

Con las pruebas realizadas se demostró que la solución es eficiente y confiable en su operación:

- Eficiencia Operacional: Se logró un tiempo total de ejecución <30 segundos, validando el rendimiento requerido para la secuencia crítica de servicios SAP.
- Confiabilidad Operacional: La tasa de fallos en el ciclo de vida se mantuvo inferior al 1% en los ciclos de prueba exitosos, con lo que concluimos que la plataforma funciona para la operatividad remota de dichos servidores.
- Impacto Económico: Con la cuantificación en los casos de estudio se demostró una optimización de costos >35% en los ambientes de subutilización.

Con todos estos datos podemos concluir que el sistema es funcional y resuelve el problema planteado al inicio, probando que la automatización del ciclo de vida de SAP ofrece una rentabilidad sustancial.

9. Trabajo a futuro

- Dentro del trabajo a futuro se contempla el integrar más nubes públicas sobre las que se encuentre instalado el sistema SAP para así apoyar a más clientes con la reducción de costos asociados a la renta de servidores en la nube.
- También se contempla el integrar un sistema de detección de transacciones en la memoria local de cada uno de los servidores de producción, para con esto implementar un programador de encendidos y apagados de los servidores de forma autónoma, si los perfiles de los clientes lo permiten. Para esto se debe integrar un algoritmo que detecte la inactividad de ambientes temporales.
- También se deja pendiente la integración de la gestión de fallos y rutinas de rollback.

Bibliografía

1) *https://aws.amazon.com/es/sap/solutions/s4hana/*
2) *https://d1.awsstatic.com/enterprise-marketing/SAP/sap-on-aws-pricing-guide.f4d83f891b3204aa642db824491fb6530070d2e2.pdf*
3) *https://d0.awsstatic.com/enterprise-marketing/SAP/SAP_on_AWS_Implementation_Guide.pdf*
4) *Client/Server Remote Control Administration System: Design and Implementation*
5) *https://www.usenix.org/legacy/event/lisa98/full_papers/yang/yang.pdf*
6) *https://www.manageengine.com/latam/network-monitoring/gestion-de-servidores.html*
7) *https://aws.amazon.com/es/blogs/awsforsap/simplify-operations-for-aws-launch-wizard-for-sap-deployments-using-aws-systems-manager-for-sap/*
8) *https://www.novis.com.mx/blog/sap/que-es-sap-basis-11292/#Historia_y_evolucion_de_SAP_Basis*
9) *https://docs.aws.amazon.com/es_es/systems-manager/latest/userguide/what-is-systems-manager.html*
10) *Smith, R. (2017). Docker orchestration.*
11) *Colino, M. P., Gomez, P. I., & McCarty, S. (2021). Red Hat Enterprise Linux 8 Administration: Master Linux Administration Skills and Prepare for the RHCSA Certification Exam. Packt Publishing.*
12) *Limoncelli, T. A., Limoncelli, T., Hogan, C. J., & Chalup, S. R. (2007). The Practice of System and Network Administration. Pearson Education.*
13) *Tanenbaum, A. S., & Van Steen, M. (2007). Distributed systems: Principles and Paradigms. Prentice Hall.*
14) *Erl, T., Puttini, R., & Mahmood, Z. (2013). Cloud computing: Concepts, Technology & Architecture. Pearson Education.*
15) *Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating system concepts.*
16) *Newman, S. (2015). Building microservices. Oreilly & Associates Incorporated.*

17) *Richardson, C. (2018). Microservices patterns: With examples in Java. Manning Publications.*
18) *SAProuter and SAP Solution Manager - General SAP Guides. (s. f.). https://docs.aws.amazon.com/es_es/sap/latest/general/overview-router-solman.html*
19) *loading. . . | SAP Help Portal. (s. f.-b). https://help.sap.com/docs/SOFTWARE_PROVISIONING_MANAGER/30839dda13b2485889466316ce5b39e9/c8ed609927fa4e45988200b153ac63d1.html*
20) *loading. . . | SAP Help Portal. (s. f.-c). https://help.sap.com/docs/SOFTWARE_PROVISIONING_MANAGER/30839dda13b2485889466316ce5b39e9/a6ca44ebebfa48da816c9baec45245f3.html*
21) *Deland-Han. (2024, 22 agosto). Descripción del protocolo de escritorio remoto (RDP) - Windows Server. Microsoft Learn. https://learn.microsoft.com/es-es/troubleshoot/windows-server/remote/understanding-remote-desktop-protocol*
22) *QuinnRadich. (2023, 13 junio). Protocolo de escritorio remoto - Win32 apps. Microsoft Learn. https://learn.microsoft.com/es-es/windows/win32/termserv/remote-desktop-protocol?source=recommendations#features*
23) *¿Qué es Python? - Explicación del lenguaje Python - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/python/*
24) *Welcome to Python.org. (2024, 7 septiembre). Python.org. https://www.python.org/about/*
25) *Holmes, L. (2008). Windows PowerShell Pocket Reference. «O'Reilly Media, Inc.»*
26) *Robinharwood. (2023, 3 febrero). PowerShell. Microsoft Learn. https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/powershell*
27) *Software de diagramación y creación de diagramas de flujo | Microsoft Visio. (s. f.). https://www.microsoft.com/es-mx/microsoft-365/visio/flowchart-software*
28) *AWS SDK para Python. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/sdk-for-python/*
29) *Boto. (s. f.). GitHub - boto/boto3: AWS SDK for Python. GitHub. https://github.com/boto/boto3*
30) *Boto3 1.35.14 documentation. (s. f.). https://boto3.amazonaws.com/v1/documentation/api/latest/index.html*
31) *Mueller, J. P. (2010). Windows Command Line Administration Instant Reference. John Wiley & Sons.*
32) *https://www.rs.uky.edu/department/feedback/Form/Sublime-Text-Power-User-Wes+Irving+Bos-1.3.pdf*
33) *Murray, K. (2011). Microsoft Office 365: Connect and Collaborate Virtually Anywhere, Anytime. O'Reilly Media, Inc.*
34) *Monk, E. F., & Wagner, B. J. (2012). Concepts in Enterprise Resource Planning (4th ed.). Cengage Learning*
35) *O'Brien, J. A., & Marakas, G. M. (2011). Management Information Systems (10th ed.). McGraw-Hill.*
36) *Patel, M. (2015). SAP ERP Financials: Configuration and Design. SAP Press.*

37) *Rodríguez, M. (2017). Gestión de pequeñas empresas: Manual práctico para microempresarios. Editorial Alfaomega.*
38) *OXXO® | Quiénes somos. (s. f.). https://www.oxxo.com/quienes-somos*
39) *IDE de Visual Studio 2022: herramienta de programación para desarrolladores de software. (2024, 12 diciembre). Visual Studio. https://visualstudio.microsoft.com/es/vs/*
40) *Garfinkel, S., & Spafford, G. (2003). *Practical UNIX and Internet Security*. O'Reilly Media.*
41) *Barrett, D., King, R., & Bynum, M. (2010). *Linux Pocket Guide*. O'Reilly Media.*
42) *Lista de empresas que usan SAP en México (1,474) | TheirStack.com. (s. f.). TheirStack.com. https://theirstack.com/es/technology/sap/mx*
43) *Herron, D. (2018). Node.Js Web Development. Packt Publishing.*
44) *Banks, A., & Porcello, E. (2017). Learning react: Functional Web Development with React and Redux. O'Reilly Media.*
45) *Coronel, C., & Morris, S. (2019). Database Systems: Design, Implementation, & Management (13th ed.). Cengage Learning.*
46) *https://docs.aws.amazon.com/pdfs/sap/latest/general/general.pdf*
47) *DeerDev. (2020, 1 noviembre). Modelo cliente servidor. https://deer.dev/blog/cliente-servidor*
48) *Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson.*
49) *Amazon Web Services (AWS). (2024). SAP on AWS Technical Documentation. AWS Whitepapers. Recuperado de https://news.sap.com/latinamerica/2024/05/aws-y-sap-abren-nuevas-vias-de-innovacion-con-ia-generativa/.*