Documento Plataforma WEB en IAAS

Juan Pablo Arias Aguirre, Omar David Bernal y Miguelangel Perez

Facultad de sistemas, Universidad Tecnológica de Pereira

Sistemas Distribuidos

Profesor: Felipe Gutierrez Isaza

27 de septiembre de 2025

Resumen.

El proyecto consistió en el diseño e implementación de una arquitectura para una plataforma de comercio electrónico. El objetivo principal fue demostrar la fragmentación horizontal de datos (Sharding), la replicación para tolerancia a fallos y la arquitectura de microservicios utilizando contenedores de sistema (Incus) sobre Linux.

Infraestructura y Virtualización.

A diferencia de soluciones tradicionales como Docker, se optó por Incus para desplegar contenedores de sistema. Esto permitió simular una red de servidores independientes (VPS) dentro de una sola máquina física, cada uno con su propia pila de red, sistema de archivos y sistema operativo completo (Ubuntu 22.04).

Inventario de Nodos (Contenedores):

Se aprovisionan 6 servidores virtuales:

- auth-server: Microservicio dedicado a la gestión de identidad (Login/Registro).
- web-dashboard: Servidor de aplicaciones principal (Frontend y Orquestador).
- db-usuarios: Base de datos exclusiva para credenciales.
- db-productos-1: Fragmento (Shard) 1 para la categoría "Tecnología".
- db-productos-2: Fragmento (Shard) 2 para categorías "Hogar/Ropa".
- db-replica: Nodo de respaldo que almacena una copia íntegra de todos los datos.

Configuración de la Capa de Datos.

Se utilizó PostgreSQL como motor de base de datos relacional debido a su robustez y cumplimiento ACID.

Por defecto, PostgreSQL solo acepta conexiones locales. Para permitir la comunicación entre contenedores a través de la red virtual de Incus, se realizó la siguiente configuración en cada nodo de base de datos:

- postgresql.conf: Se modificó la directiva listen_addresses = '*.' para escuchar en todas las interfaces de red.
- pg_hba.conf: Se configuró la regla host all all 0.0.0.0/0 md5 para permitir conexiones autenticadas desde el servidor web y el servidor de autenticación.

Arquitectura de Software y Lógica de Negocio.

Microservicio de Autenticación.

Para desacoplar la seguridad de la lógica de negocio, se creó un servidor API independiente en Python (Flask).

Seguridad: Las contraseñas se almacenan como Hashes (PBKDF2) y no en texto plano.

Sesiones: Se implementó el estándar JWT (JSON Web Tokens) para manejar la sesión del usuario de forma stateless (sin estado).

Estrategia de Sharding (Fragmentación).

El núcleo del proyecto es la distribución inteligente de datos. Se implementó un Directory-Based Sharding a nivel de aplicación (en el código Python del Dashboard).

Lógica: Al crear un producto, el sistema evalúa su categoría.

- Si es Tecnología, se envía la escritura a la IP de db-productos-1.
- Si es Hogar/Ropa, se envía la escritura a la IP de db-productos-2.

Beneficio: Esto reduce la carga de escritura y lectura en un solo servidor, distribuyendo el tráfico horizontalmente.

Replicación por Doble Escritura (Dual Write).

Para garantizar la redundancia de datos, se implementó un mecanismo de replicación síncrona a nivel de código.

Funcionamiento: Cada vez que el servidor recibe una petición de INSERT o DELETE, ejecuta la transacción primero en el nodo principal correspondiente (Shard) y, acto seguido, replica la misma instrucción en el nodo db-replica.

Consistencia: Esto asegura que, en caso de fallo de uno de los nodos principales, exista una copia de seguridad actualizada en un servidor independiente.

Interfaz y Experiencia de Usuario.

Backend: Desarrollado en Python con el framework Flask y la librería psycopg2 para la conectividad de base de datos.

Frontend: Se diseñó un Dashboard utilizando HTML5 y Bootstrap 5, permitiendo visualizar de forma clara en qué nodo reside cada dato (mediante etiquetas de colores) y realizar operaciones CRUD completas.

Gestión y Acceso Remoto.

Incus UI: Se intentó habilitar la interfaz de administración gráfica Incus UI en el puerto 8443 siguiendo los procedimientos oficiales de exposición del servicio y generación de tokens de confianza. Sin embargo, se identificó una limitación técnica en el entorno virtualizado WSL2, donde las políticas de seguridad de los navegadores en Windows bloquearon sistemáticamente el intercambio de certificados cliente-servidor necesarios para la autenticación en localhost. Debido a esta restricción de red y para mantener la integridad de la seguridad del sistema sin alterar los certificados raíz del host, se optó por realizar la administración integral del clúster mediante la Interfaz de Línea de Comandos, utilizando herramientas nativas como incus top e incus list, lo cual garantiza un monitoreo eficiente de recursos y se alinea con los estándares de gestión de servidores en producción.

Acceso Externo (Ngrok): Para las pruebas de validación remota, se implementó un túnel seguro con Ngrok, exponiendo el puerto 8080 del contenedor web a internet, permitiendo el acceso y auditoría del sistema desde redes externas (móviles/remotas).

Conclusión.

El sistema resultante es una arquitectura distribuida funcional que cumple con los principios de modularidad, escalabilidad horizontal (mediante Sharding) y alta disponibilidad de

datos (mediante Replicación). Se superaron desafíos complejos de enrutamiento de red en entornos WSL2 y sincronización de datos entre múltiples instancias de bases de datos."