Valida tu ID

Cárdenas, Yaxul
Delgado, Juan
Díaz, Yesenia
Martinez, Gabriel
Ing. de Datos e Inteligencia Artificial
Universidad Autonoma del Occidente
Cali, Colombia

Abstract

The Validate Your ID system is developed to verify identification for financial product applications virtually using Docker and Vagrant. This article describes the system architecture, user roles, types of requests, request statuses, and system configuration in detail.

El sistema Valida tu ID se desarrolla para comprobar la validación de identificaciones para solicitudes de productos financieros de manera virtual utilizando Docker y Vagrant. Este artículo describe la arquitectura del sistema, los roles de usuario, los tipos de solicitudes, los estados de solicitud y la configuración del sistema en detalle.

Keywords: Identity Validation, Docker, Vagrant, Information System

I. Introducción

El sistema Valida tu ID es una solución diseñada para la validación de identidades mediante la gestión de solicitudes en un entorno controlado. Este sistema se despliega utilizando tecnologías modernas como Docker y Vagrant para asegurar una configuración y administración eficientes. La validación de identidades es crucial en diversos contextos, como el acceso a servicios financieros, la verificación de usuarios en plataformas en línea y el cumplimiento de normas y regulaciones. Este sistema facilita la gestión de solicitudes a través de diferentes roles de usuario y estados de solicitud, proporcionando una estructura clara y robusta para el procesamiento de datos.

II. Sistema de roles

El sistema está diseñado para manejar varios roles, cada uno con permisos específicos que definen sus capacidades dentro del sistema.

A. *Administrador: Este rol tiene acceso completo para gestionar el sistema. Las responsabilidades incluyen la creación y gestión de usuarios, así como la supervisión de todas las peticiones. El administrador asegura que el sistema funcione correctamente y que todas las solicitudes sean procesadas eficientemente.*

B. *Validador: Los validadores son responsables de revisar y procesar las solicitudes de los clientes. Pueden cambiar el estado de las solicitudes de "Pendiente" a "Aprobado" o "Rechazado", según la conformidad de las solicitudes con los criterios establecidos. Este rol es crucial para asegurar que solo las solicitudes válidas sean aprobadas.*

C. *Cliente: Los clientes son los usuarios que solicitan servicios a través del sistema. Pueden realizar solicitudes para la validación de sus identidades y seguir el estado de sus peticiones. Este rol permite la interacción directa con el sistema para la presentación de solicitudes y seguimiento de su progreso.*

III. Tipos de peticiones

Los clientes tienen la posibilidad de realizar solicitudes de servicios a través de una interfaz intuitiva y fácil de usar. Los tipos de solicitudes disponibles son:

A. *Certificado de Depósito a Término (CDT): Esta opción permite a los clientes solicitar la creación y gestión de CDs, ofreciendo una forma segura y eficiente de invertir sus fondos a plazo fijo con un interés determinado.*

B. *Cuenta de Ahorros: Los clientes pueden abrir o gestionar sus cuentas de ahorros, facilitando la administración de sus finanzas personales y el acceso a sus fondos de manera rápida y segura.*

C. *Crédito: Esta solicitud está destinada a la gestión de productos crediticios, permitiendo a los clientes solicitar préstamos o líneas de crédito para diversas necesidades financieras, con un proceso claro y estructurado para la aprobación de dichos productos.*

Cada tipo de solicitud está diseñado para cumplir con los estándares del sistema, asegurando una experiencia de usuario óptima y eficiente en la gestión de sus servicios financieros.

IV. Estados de las peticiones

En el sistema "Valida tu ID", las solicitudes de servicios financieros pueden encontrarse en diferentes estados a lo largo de su proceso de validación. Cada estado refleja una etapa en el ciclo de vida de la solicitud, proporcionando transparencia tanto para los validadores como para los clientes. A continuación, se detallan los posibles estados y sus implicaciones:

A. *Pendiente: Este es el estado inicial de todas las solicitudes al momento de ser creadas por el cliente. Una solicitud en estado "Pendiente" está a la espera de ser revisada por un validador. En esta etapa, el cliente puede visualizar que su solicitud ha sido recibida, pero aún no ha sido procesada.*

B. *Aprobado: Una vez que la solicitud ha sido revisada y cumple con todos los criterios y requerimientos, se marca como "Aprobado". Este estado indica que la solicitud ha pasado todas las verificaciones necesarias y el cliente recibirá el servicio solicitado o el producto financiero que solicitó. La aprobación implica que el proceso ha finalizado exitosamente y que el cliente puede proceder con los siguientes pasos según el tipo de solicitud (por ejemplo, abrir una cuenta, activar un CDT, etc.).*

C. *Rechazado: Si la solicitud no cumple con los criterios establecidos o presenta inconsistencias, el estado cambiará a "Rechazado". Este estado indica que la solicitud ha sido evaluada y no puede ser aprobada debido a errores, faltas de conformidad, o documentación insuficiente. El cliente podrá visualizar este estado junto con posibles motivos de rechazo, lo que permite realizar las correcciones necesarias en futuras solicitudes.*

V. Arquitectura

El sistema Valida tu ID utiliza una arquitectura de microservicios, donde cada componente del sistema está desacoplado y es independiente, lo que facilita la escalabilidad y el mantenimiento. Los microservicios se comunican entre sí a través de interfaces bien definidas, y cada uno se ejecuta en su propio contenedor Docker.

A. *Diagrama de Despliegue*

El diagrama de despliegue del sistema muestra cómo se distribuyen los componentes del sistema en diferentes servidores, asegurando una operación robusta y escalable. El sistema se despliega en dos servidores Ubuntu configurados como un clúster de Docker.

El diagrama de despliegue ilustra la distribución de estos componentes en los servidores del clúster Docker:

1. Servidor 1 (servidorUbuntu): Aloja los microservicios y la base de datos. Este servidor maneja la mayor parte del procesamiento de datos y la lógica de negocio del sistema.
2. Servidor 2 (clienteUbuntu): Aloja el portal de usuario y los balanceadores de carga. Este servidor se encarga de distribuir las solicitudes entrantes entre los diferentes servicios y proporcionar una interfaz de usuario accesible.

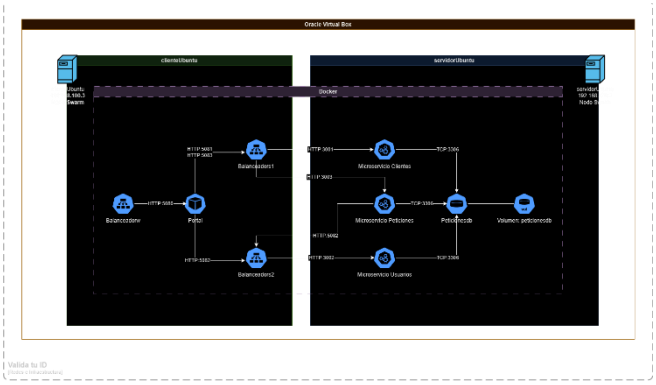

Fig. 1. Diagrama de flujo de despliegue

B. *En un diagrama de capas, nuestro sistema se ve de la siguiente forma:*

1. Capa de Presentación (Frontend): Incluye el portal de usuario que interactúa con los clientes. Esta capa está diseñada para ser intuitiva y fácil de usar, proporcionando una interfaz gráfica para la gestión de solicitudes.
2. Capa de Servicios (Backend): Aquí se encuentran los microservicios responsables de las funcionalidades del sistema, como microclientes, micropeticiones y microusuarios. Estos servicios manejan la lógica de negocio y las operaciones específicas del sistema.
3. Capa de Datos (Database): La base de datos Peticionesdb almacena toda la información relacionada con las solicitudes, usuarios y clientes. Esta capa asegura la integridad y disponibilidad de los datos mediante el uso de un sistema de gestión de bases de datos relacional..

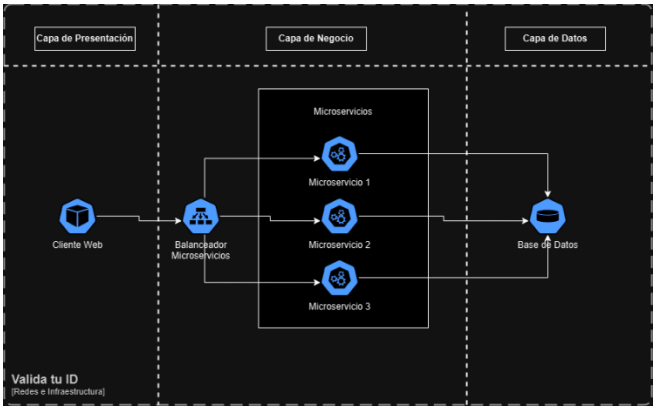

Fig. 2. D iagrama lógico y capas del sistema

VI. Servicios de Despliegue

El sistema despliega varios servicios, cada uno con una función específica, asegurando la separación de responsabilidades y la facilidad de mantenimiento.

TABLA I. Servicios de Despliegue

Cada uno de estos servicios se despliega de manera independiente y se comunica a través de un balanceador de carga, lo que permite una distribución eficiente del tráfico y la escalabilidad del sistema.

Servicio	Servicios de despliegue		
	Dirección directa de URL	***Balanceador***	***Dirección URL del balanceador***
Portal	http://192.168.100.2:3000	Balanceadorw	http://192.168.100.2:5080/
Micro - clientes	http://192.168.100.2:3001	Balanceadors 1	http://192.168.100.2:5081/
Micro - peticiones	http://192.168.100.2:3003	Balanceadors 1	http://192.168.100.2:5083/
Micro - usuarios	http://192.168.100.2:3002	Balanceadors 2	http://192.168.100.2:5082/
Peticiones db	No aplica	No aplica	No aplica

VII. Configuración de la base de datos

El sistema utiliza una base de datos MySQL para almacenar la información de usuarios, clientes y peticiones. La configuración de la base de datos es crucial para asegurar la integridad y disponibilidad de los datos.

1) *Base de Datos: Peticionesdb*
2) *Puerto del Contenedor: 3306*
3) *Puerto Host: 32000*

A. Tabla de usuarios

Registro	Tabla de usuarios		
	Campo	***Descripción***	***Valores***
1	usuario	Nick Name o nombre del usuario	
2	nombre	Nombre completo de a quién pertenece el usuario	
3	rol	Rol que tiene este usuario en el sistema	
4	password	Contraseña del usuario	

TABLA II. BASE DE DATOS, TABLA USUARIOS

B. Tabla de clientes

Registro	Tabla de clientes		
	Campo	***Descripción***	***Valores***
1	cc	Número de documento	
2	password	Clave de acceso al portal	

TABLA III. BASE DE DATOS, TABLA CLIENTES

C. Tabla de peticiones

Registro	Tabla de peticiones		
	Campo	***Descripción***	***Valores***
1	id	ID del registro de la petición	
2	ccclliente	Identificación del cliente	
3	ccarchivo	Ruta del archivo cargado	
4	tiposervicio	Tipo de Servicio para la petición	
5	bancocliente	Banco para la solicitud	
6	fechasolicitud	Fecha del registro de la solicitud	
7	horasolicitud	Hora del registro de la solicitud	
8	usuariovalidador	Usuario que validó la solicitud	
9	nombrevalidador	Nombre del usuario que validó la solicitud	
10	estado	Estado de la solicitud	
11	fechacreacion	Fecha de creación de la petición	
12	horacreacion	Hora de creación de la petición	
13	fecharevision	Fecha de revisión de la petición	
14	horarevision	Hora de revisión de la petición	

TABLA IV. BASE DE DATOS, TABLA PETICIONES

D. Diagrama de flujo

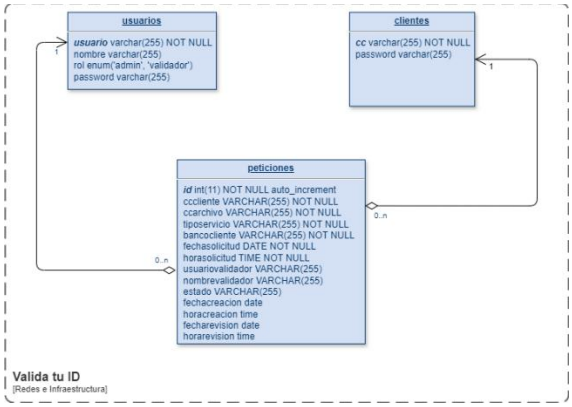

Fig. 3. Diagrama de flujo de la base de datos

VIII. Instalación del sistema

La instalación del sistema incluye la creación de máquinas virtuales, la instalación de Docker, la descarga del repositorio y la construcción de las imágenes Docker.

A. Creación de maqina virtual

Las máquinas virtuales se crean usando Vagrant con la siguiente configuración:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config|
 if Vagrant.has_plugin? "vagrant-vbguest"
  config.vbguest.no_install = true
  config.vbguest.auto_update = false
  config.vbguest.no_remote = true
 end
 config.vm.define :servidorUbuntu do |servidorUbuntu|
  servidorUbuntu.vm.box = "bento/ubuntu-22.04"
 servidorUbuntu.vm.network :private_network, ip: "192.168.100.2"
  servidorUbuntu.vm.hostname = "servidorUbuntu"
  servidorUbuntu.vm.box_download_insecure=true
  servidorUbuntu.vm.provider "virtualbox" do |vb|
   vb.memory = "2048"
   vb.cpus = 2
  end
 end
 config.vm.define :clienteUbuntu do |clienteUbuntu|
  clienteUbuntu.vm.box = "bento/ubuntu-22.04"
  clienteUbuntu.vm.network :private_network, ip: "192.168.100.3"
  clienteUbuntu.vm.hostname = "clienteUbuntu"
  clienteUbuntu.vm.box_download_insecure=true
  clienteUbuntu.vm.provider "virtualbox" do |vb|
   vb.memory = "2048"
   vb.cpus = 2
  end
 end
end
```

B. Instalación de Docker

Instale Docker en ambos servidores, servidorUbuntu y clienteUbuntu.

```
# Actualizar el sistema: Primero, actualiza la base de datos de paquetes con el siguiente comando:
sudo apt update
# Instalar Docker: Instala Docker Engine utilizando el siguiente comando:
sudo apt install docker.io
# Habilitar Docker: Habilita el servicio de Docker para que se inicie automáticamente:
sudo systemctl enable docker
# Verificar el estado de Docker: Asegúrate de que el servicio de Docker esté en ejecución:
sudo systemctl status Docker
# Probar Docker: Ejecuta un contenedor de prueba para verificar la instalación:
sudo docker run hello-world
# Repetir proceso para la otra máquina virtual (ya sea servidorUbuntu o clienteUbuntu)
```

C. Descargue de repositorio y construción de imágenes de Docker

Descargue el repositorio y construya las imágenes Docker:

```
git clone https://github.com/jacoboDM/ValidaTuID
cd ValidaTuID/CreacionImagenes
docker compose build
docker images -a
```

D. Publicación de imágenes de Docker

Publique las imágenes en Docker Hub:

```
sudo docker login -u Usuario1
docker tag proyectofinal-balanceadorw Usuario1/proyectofinal-balanceadorw
sudo docker push Usuario1/proyectofinal-balanceadorw
docker tag proyectofinal-portal Usuario1/proyectofinal-portal
sudo docker push Usuario1/proyectofinal-portal
```

E. Configuración de archivo fuente

Edite el archivo docker-compose.yml para apuntar a las imágenes en Docker Hub, cambiando “Usuario1” por el nombre de usuario suyo de Docker Hub.

F. Construcción del sistema

Ejecutar el siguiente comando dentro de la carpeta CreacionImagenes del repositorio

```
docker compose up --build
```

G. Configuración de roles de despliegue

Configure los roles de despliegue en el clúster:

```
docker node update --label-add role=servidor servidorUbuntu
docker node update --label-add role=cliente clienteUbuntu
```

H. Creación del cluster en Docker Swarm.

Cree el clúster con Docker Swarm:

```
docker stack deploy -c docker-compose.yml proyectofinal
docker stack ls
docker service ls
```

IX. Pruebas de rendimiento

1) Archivo de JMeter: Asegúrate de tener el archivo Pruebas de Carga.jmx *configurado para la consulta de todos los usuarios de la base de datos.*

2) Escalar el Servicio: Para escalar el servicio y evitar fallos durante la prueba de carga, ejecuta el siguiente comando en servidorUbuntu:

```
docker service scale proyectofinal_microusuarios=2
```

A. Estadísticas del Balanceador de Carga

1) Ver Estadísticas: Para ver las estadísticas de las peticiones a través de los balanceadores, utiliza las siguientes URLs:

Estadística	Balanceador		
	Servicio	***URL***	***Descripción***
1	Balanceadorw	http://192.168.100.2:5080/haproxy?stats	Estadísticas del Balanceador del Portal

TABLA V. ESTADISTICAS DE CARGA EN LOS BALANCEADORES

X. RESULTADO DE ANALISIS

Para evaluar el rendimiento y la efectividad del sistema Valida tu ID, hemos implementado un conjunto de análisis utilizando Apache Spark, PySpark, archivos CSV, y la base de datos del sistema. Estos análisis se presentan en un dashboard interactivo desplegado en Docker, permitiendo una visualización clara y precisa de los datos.

A. Cantidad de Peticiones por Estado:

Este análisis muestra la distribución de las solicitudes en diferentes estados: Aprobado, Rechazado y Pendiente. Ayuda a identificar la eficiencia del sistema en el procesamiento de solicitudes y permite monitorear el volumen de trabajo pendiente.

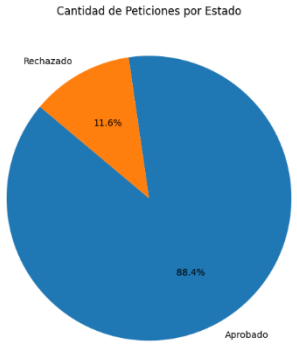

Fig. 4. Reporte de peticiones por estado.

B. Cantidad de Peticiones por Hora de Creación:

Analiza el número de solicitudes creadas en diferentes horas del día. Este reporte es útil para identificar patrones de uso y periodos de alta demanda, lo cual puede ayudar en la optimización de los recursos del sistema.

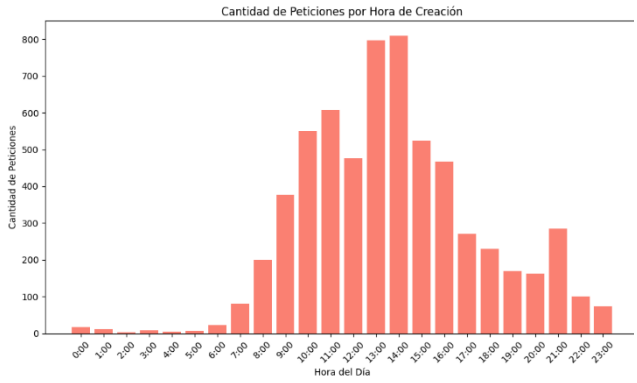

Fig. 5. Reporte de peticiones por hora de creación.

C. Cantidad de Peticiones por Tipo de Servicio:

Desglosa las solitudes según los tipos de servicios (CDT, Cuenta de Ahorros y Crédito). Permite evaluar cuál es el servicio más demandado y ajustar las estrategias de servicio en consecuencia.

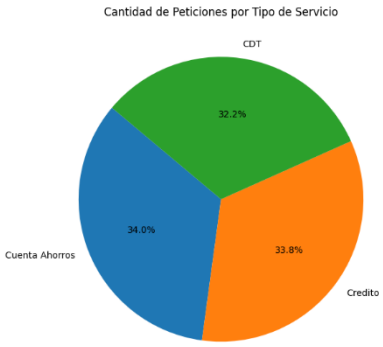

Fig. 6. Reporte de peticiones por hora de creación.

D. Cantidad de Peticiones por Usuario:

Muestra el número de solicitudes realizadas por cada usuario. Este reporte ayuda a identificar a los usuarios más activos y a comprender mejor la distribución de la carga de trabajo entre los validadores.

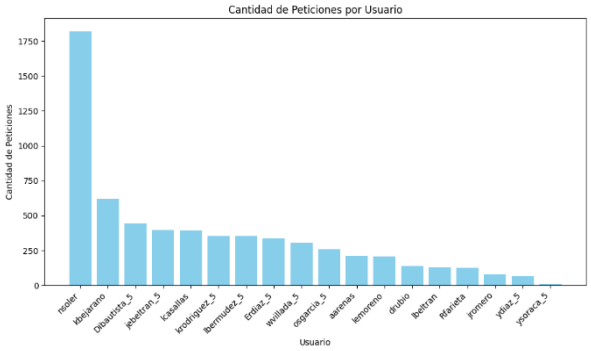

Fig. 7. Reporte de peticiones por usuario.

E. Top 10 Clientes con Más Peticiones:

Identifica a los 10 clientes con el mayor número de solicitudes. Este análisis es crucial para enfocarse en los clientes más activos y potencialmente brindarles un servicio más personalizado.

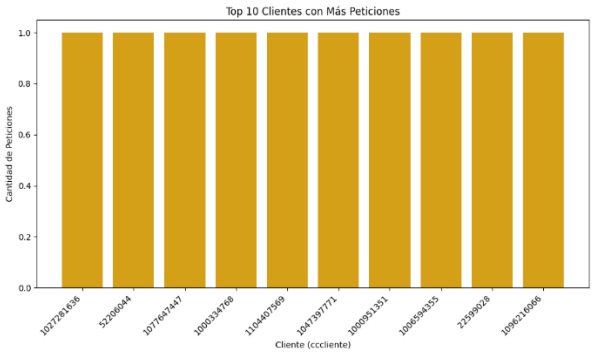

Fig. 8. Reporte de los top 10 clientes con más peticiones.

F. Tiempo de Respuesta de los Validadores:

Monitorea el tiempo promedio que tardan los validadores en procesar una solicitud desde que se crea hasta que se aprueba o rechaza. Este reporte es esencial para identificar posibles cuellos de botella y mejorar la eficiencia del sistema.

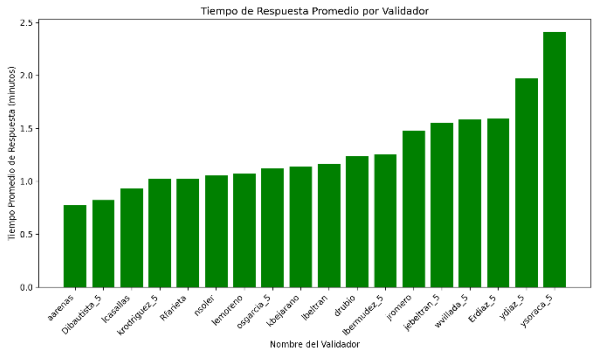

Fig. 9. Reporte de tiempo de respuesta de los validadores

G. Visualización en el Dashboard:

Todos los análisis se presentan en un dashboard interactivo, que ofrece las siguientes características:

1) Filtros Dinámicos: Permiten a los usuarios personalizar las vistas de los reportes según diferentes criterios como fechas, tipos de servicio y estados de solicitud.

2) Gráficos Interactivos: Facilitan la comprensión de los datos a través de gráficos de barras, líneas y pasteles.

REFERENCES

[1] A. Brown et al., "Designing Scalable Information Systems," Journal of Systems Architecture, vol. 75, pp. 45-58, Jan. 2020.

[2] A. Deacon and M. Humphrey, "Access Control and Delegation in Distributed Systems," IEEE Internet Computing, vol. 7, no. 3, pp. 40-47, May 2003.

[3] "Apache JMeter." Apache Software Foundation, https://jmeter.apache.org/, accessed November 7, 2024.

[4] "Apache Spark." Apache Software Foundation, https://spark.apache.org/docs/latest/, accessed November 7, 2024.

[5] C. Coronel, S. Morris, and P. Rob, "Database Systems: Design, Implementation, & Management," 13th ed., Cengage Learning, 2018.

[6] D. Martin, "Workflow Management: Models, Methods, and Systems," MIT Press, 2002.

[7] "Docker Hub: The World's Largest Library and Community for Container Images." Docker, https://hub.docker.com/, accessed November 7, 2024.

[8] "Docker Swarm: Simple Docker Orchestration Guide," YouTube, 11-Sep-2016. [Online]. Available: https://www.youtube.com/watch?v=cqbh-RneBlk. [Accessed: 07-Nov-2024].

[9] "Information Dashboard Design: Displaying Data for At-a-Glance Monitoring," 2nd ed., Analytics Press, 2013.

[10] "Install Docker Engine on Ubuntu." Docker Documentation, https://docs.docker.com/engine/install/ubuntu/, accessed November 7, 2024.

[11] G. K. Gupta, "Database Management Systems," Tata McGraw-Hill, 2009.

[12] "Get Started with Docker Swarm." Docker Documentation, https://docs.docker.com/engine/swarm/, accessed November 7, 2024.

[13] J. Smith, "Techniques for Identity Verification in Digital Systems," IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1234-1245, May 2018.

[14] J. Watson, "E-Government and E-Services: Concepts, Methodologies, Tools, and Applications," IGI Global, 2010.

[15] K. Hightower, B. Burns, and J. Beda, "Kubernetes: Up and Running," O'Reilly Media, 2017.

[16] L. Lamport, "Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers," Addison-Wesley, 2002.

[17] M. Mulvey, "Financial Services: Understanding and Implementing Emerging Trends," McGraw-Hill, 2015.

[18] M. Richards, "Microservices Patterns: With examples in Java," Manning Publications, 2018.

[19] "MySQL Reference Manual." Oracle, https://dev.mysql.com/doc/, accessed November 7, 2024.

[20] "Overview of Docker Compose." Docker Documentation, https://docs.docker.com/compose/overview/, accessed November 7, 2024.

[21] "PySpark Documentation." Apache Spark, https://spark.apache.org/docs/latest/api/python/index.html, accessed November 7, 2024.

[22] R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems," 2nd ed. Wiley, 2008.

[23] S. Newman, "Building Microservices: Designing Fine-Grained Systems," O'Reilly Media, 2015.

[24]

[25] "Scale Services in Swarm." Docker Documentation, https://docs.docker.com/engine/swarm/services/#scale-a-service, accessed November 7, 2024.

[26] T. Erl, "Cloud Computing: Concepts, Technology & Architecture," Prentice Hall, 2013.

[27] "Vagrant Documentation." HashiCorp, https://www.vagrantup.com/docs, accessed November 7, 2024