**UNIVERSIDAD DE LAS FUERZAS ARMADAS “ESPE”**



**APLICACIÓN DE SISTEMAS OPERATIVOS**

**PROYECTO UNIDAD 2**

**CEVALLOS JUAN**

**QUIMUÑA KEVIN**

**PACHACAMA FREDDY LEONEL**

**SANGOQUIZA DAVID**

**NRC: 10035**

1. **Objetivos**
   1. **Objetivo General**

* El objetivo de este trabajo es implementar una arquitectura basada en microservicios utilizando contenedores Docker para optimizar y mejorar la, flexibilidad y eficiencia de un modelo de negocio específico.
  1. **Objetivos Específicos**
* Implementar una arquitectura basada en microservicios dentro de contenedores Docker para la página web del modelo de negocio.
* Crear un servicio de asistente virtual inteligente que se integre con la página web implementada en Docker.
* Configurar y desplegar una base de datos distribuida en contenedores Docker, optimizada para el almacenamiento y recuperación eficiente de los datos requeridos por los distintos microservicios, usando PHP.

1. **Introducción**

En los últimos años, la evolución tecnológica ha impulsado la transformación digital de numerosas empresas y organizaciones, llevándolas a adoptar arquitecturas de software más flexibles y escalables. En este contexto, los microservicios han surgido como una arquitectura de desarrollo de aplicaciones que ha revolucionado la forma en que se construyen, implementan y mantienen los sistemas informáticos.

Los microservicios son una metodología de diseño que consiste en dividir una aplicación monolítica en pequeños servicios independientes, cada uno enfocado en tareas específicas y comunicándose entre sí mediante APIs bien definidas. Esta aproximación permite que los equipos de desarrollo se centren en unidades más pequeñas y manejables, lo que a su vez favorece la agilidad, facilita la colaboración y mejora la capacidad de respuesta frente a cambios en los requisitos del negocio.

Sin embargo, la implementación de microservicios también plantea nuevos desafíos, como la gestión de dependencias, la escalabilidad individual de cada servicio y la garantía de un despliegue consistente y confiable. Es en este punto donde Docker, una plataforma de contenedores, se presenta como una solución poderosa y ampliamente utilizada en la comunidad de desarrollo.

Docker proporciona un entorno de ejecución ligero y aislado para los microservicios, encapsulando cada uno en su propio contenedor. Estos contenedores incluyen todo lo necesario para que el servicio funcione de manera consistente en diferentes entornos, independientemente de las diferencias de configuración y bibliotecas del sistema operativo. La portabilidad, eficiencia y velocidad que ofrece Docker se traducen en una mayor facilidad para el desarrollo, prueba y despliegue de microservicios.

En este trabajo, exploraremos en detalle el uso de Docker para implementar microservicios, analizando los beneficios y desafíos que esta combinación conlleva. Además, se presentarán ejemplos prácticos de cómo Docker permite la creación de un entorno de desarrollo y producción más confiable y escalable para los servicios que componen una aplicación empresarial. Asimismo, se destacarán las mejores prácticas y consideraciones clave para aprovechar al máximo esta poderosa herramienta en el desarrollo y mantenimiento de arquitecturas basadas en microservicios. Con esto, esperamos proporcionar una visión clara y práctica sobre cómo Docker se ha convertido en una pieza fundamental en el camino hacia aplicaciones más modulares y flexibles para satisfacer las demandas del mundo empresarial actual.

1. **Marco teórico y Materiales.**
   1. **Materiales**

* HiperVisor (VirtualBox)
* S.O de Kali Linux
* Laptop
  1. **Marco teórico**

**Docker**

Docker es una plataforma de código abierto que permite a los desarrolladores empaquetar, distribuir y ejecutar aplicaciones dentro de contenedores. Un contenedor es una unidad de software que incluye todo lo necesario para que una aplicación se ejecute de manera autónoma, incluyendo el código, las bibliotecas, las dependencias y las variables de entorno. Los contenedores Docker se basan en el principio de virtualización a nivel de sistema operativo, lo que proporciona un alto grado de aislamiento y portabilidad.

**Ventajas y Beneficios:**

**Portabilidad:** Los contenedores Docker se pueden ejecutar en cualquier entorno que tenga Docker instalado, independientemente del sistema operativo o la infraestructura subyacente. Esto garantiza que una aplicación funcione de la misma manera en desarrollo, pruebas y producción.

**Eficiencia:** Los contenedores comparten el kernel del sistema operativo, lo que los hace más livianos en comparación con las máquinas virtuales tradicionales. Esto permite un inicio más rápido y un uso más eficiente de los recursos del sistema.

**Escalabilidad:** Docker facilita la creación de múltiples instancias de contenedores que pueden escalarse de manera horizontal para manejar la carga de trabajo en aumento.

**Aislamiento:** Cada contenedor tiene su propio entorno aislado lo que evita conflictos entre las dependencias y mejora la seguridad del sistema.

**Estructura de Docker:**

**Imágenes:** Las imágenes Docker son plantillas de solo lectura que contienen el código y todas las dependencias necesarias para ejecutar una aplicación. Se utilizan para crear contenedores en tiempo de ejecución.

**Contenedores:** Los contenedores son instancias en ejecución de una imagen Docker. Cada contenedor es un entorno aislado que puede interactuar con el sistema host y otros contenedores.

**Registro:** Los registros Docker son repositorios que almacenan imágenes Docker, lo que facilita la distribución y el intercambio de imágenes entre desarrolladores y equipos.

**Microservicios**

Los microservicios son una arquitectura de desarrollo de aplicaciones que consiste en dividir una aplicación monolítica en pequeños servicios independientes y autónomos. Cada microservicio se enfoca en una tarea específica y se comunica con otros servicios a través de interfaces bien definidas, generalmente utilizando tecnologías web como HTTP/REST o gRPC. La arquitectura de microservicios promueve el modularidad, la escalabilidad, la flexibilidad y la facilidad de mantenimiento.

**Características clave de los microservicios:**

**Desacoplamiento:** Los microservicios están diseñados para ser independientes y no tienen dependencias directas entre sí. Esto permite un desarrollo y despliegue más ágil y modular.

**Escalabilidad individual:** Cada servicio puede escalarse por separado en función de sus necesidades, lo que permite un uso eficiente de los recursos y una mejor capacidad de respuesta ante picos de carga.

**Tolerancia a fallos:** Los microservicios están diseñados para ser tolerantes a fallos, lo que significa que un fallo en un servicio no debe afectar a otros servicios, lo que mejora la confiabilidad del sistema en su conjunto.

Desarrollo y despliegue independiente: Los equipos de desarrollo pueden trabajar en paralelo en diferentes microservicios, lo que permite un desarrollo más rápido y una **entrega continua.**

**Políglota:** Los microservicios no están limitados a un solo lenguaje de programación o tecnología, lo que permite a los equipos utilizar la mejor herramienta para cada servicio.

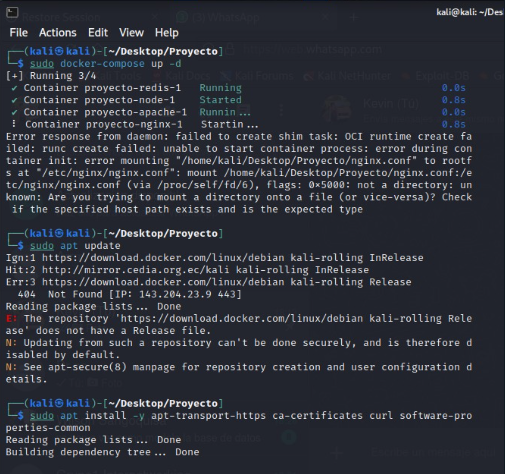
**Kubernetes;**

Kubernetes es una plataforma de código abierto y una herramienta poderosa de orquestación de contenedores que permite la gestión y escalado de aplicaciones de contenedores de manera eficiente y automatizada. Fue desarrollado por Google a partir de su experiencia interna con la gestión de contenedores a gran escala, y posteriormente fue donado a la Cloud Native Computing Foundation (CNCF) en 2015. Desde entonces, se ha convertido en la solución de orquestación más popular y ampliamente utilizada en el mundo de los contenedores.

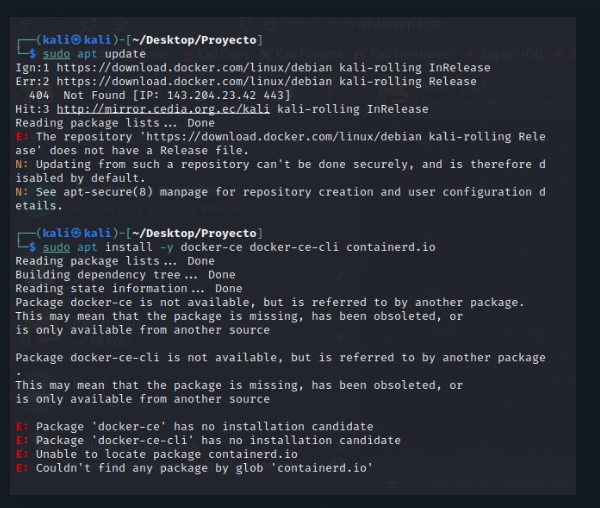
Kubernetes es una solución altamente escalable y versátil que se adapta tanto a entornos locales como a entornos en la nube. Permite a los equipos de desarrollo y operaciones implementar, administrar y escalar aplicaciones de contenedores de manera eficiente, proporcionando una infraestructura robusta y confiable para aplicaciones modernas basadas en microservicios. La comunidad en constante crecimiento, el soporte de múltiples proveedores de servicios en la nube y el compromiso con la innovación hacen que Kubernetes sea una opción sólida para el despliegue y gestión de aplicaciones en el mundo de los contenedores.

* 1. **Procedimiento**

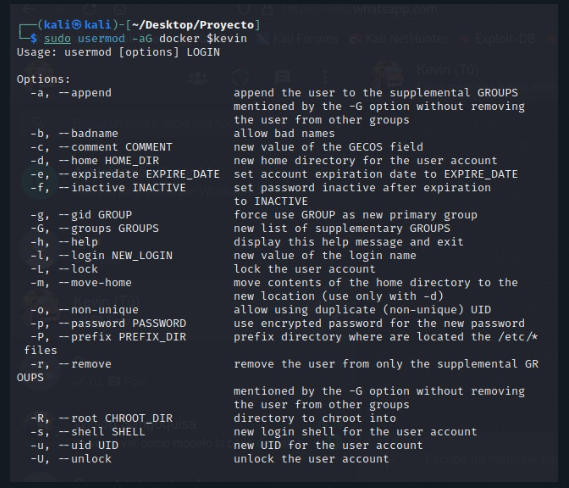
1. Se actualiza los paquetes para poder instalar los contenedores.



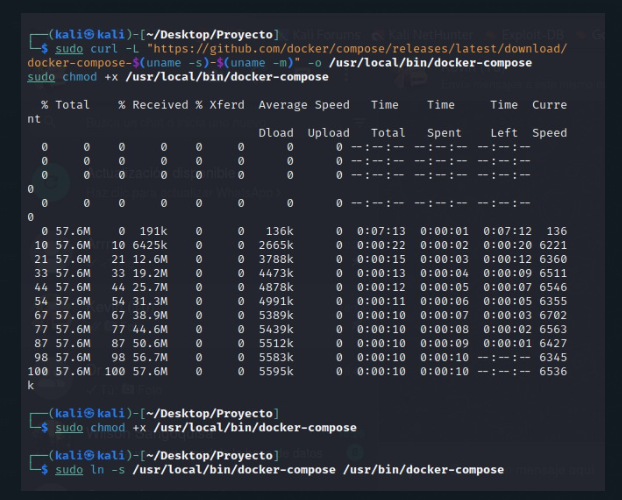
1. Instalar el sistema de Docker y agregar el repositorio



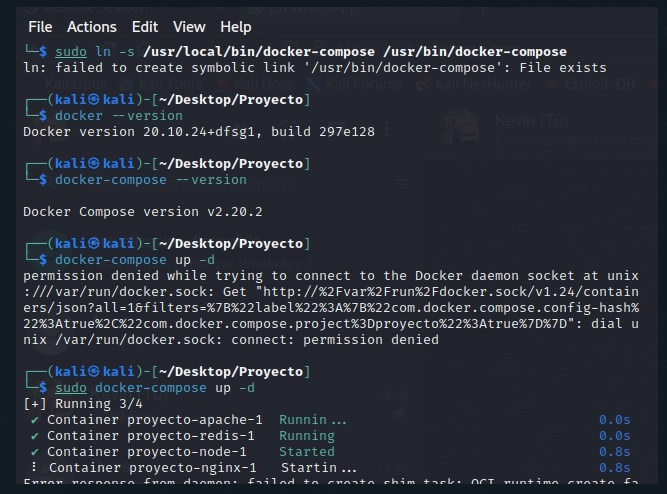
1. Agregar un usuario al grupo de trabajo para el uso de Dockers sin contraseña



1. Se descarga la última versión de Docker Compose y se aplica permisos de ejecución.

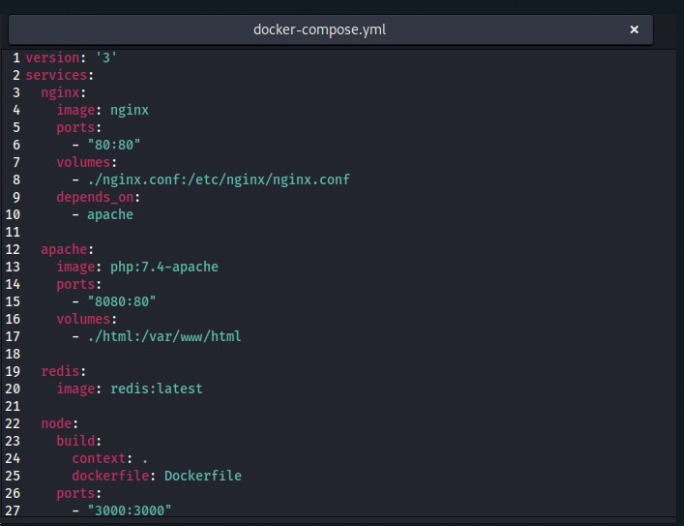


1. Se crea un enlace simbólico, para poder acceder a Docker Compose desde cualquier ubicación y se verifica las versiones de Docker y Docker Compose

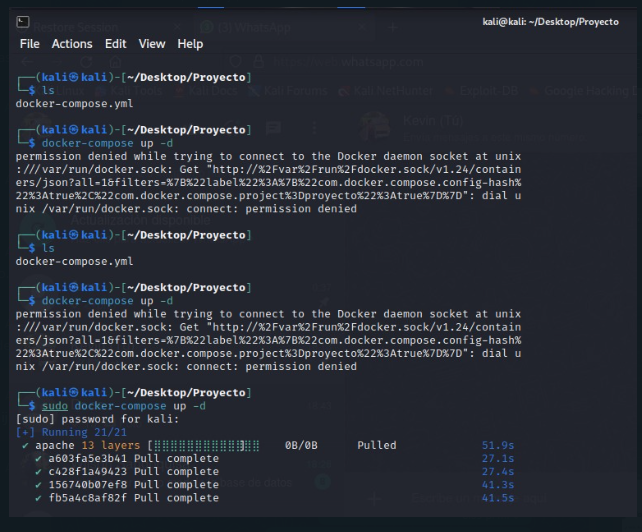


**Implementación de Nginx, Web (Asistente Virtual) APACHE y PHP**

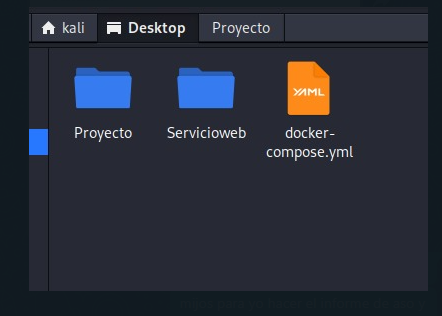
1. Se crea un archivo llamado “docker-compose yml”, en el cual se utilizan las imágenes oficiales de Nginx, Apache, PHP y Redis de Docker Hub y se mapean los puertos de los contenedores para la comunicación.

****

1. Se guarda los cambios realizados en el documento “docker-compose yml”



1. En el mismo directorio en donde esta “docker-compose yml” se crea una carpeta HTML, para poder colocar los archivos necesarios para la página web donde se encuentra nuestro asistente virtual.



1. Se ejecuta el comando “docker start $(docker ps –aq)” para iniciar todos los contenedores.

**Implementación de Redis**

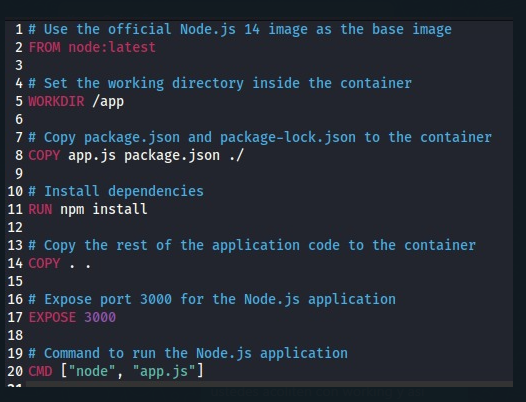
1. Se crea un archivo llamado “app.js” para agregar las funciones para poder crear un pequeño servidor Node.js que interactúe con Redis.



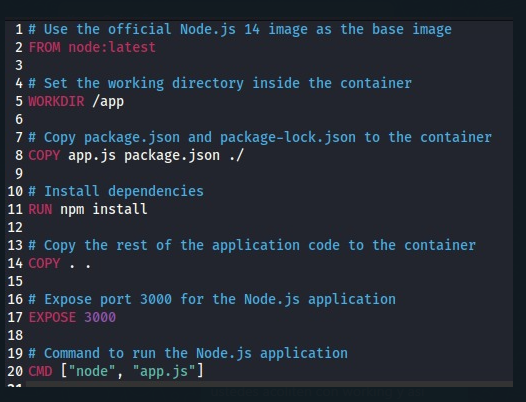
1. Se crea un archivo llamado “package. json” para las dependencias de Node.js y se instalan las dependencias express y redis.



1. Se crea un archivo llamado Dockerfile en el mismo directorio

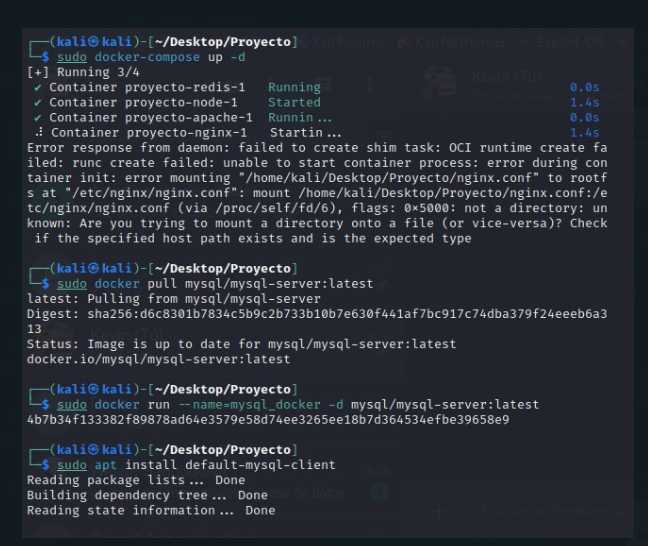


1. Se ejecuta el comando “docker-compose up –d”

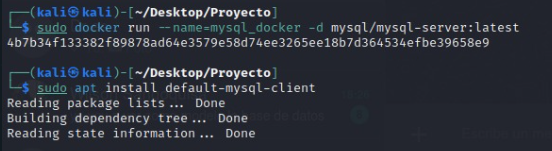


**Implementación de Mysql**

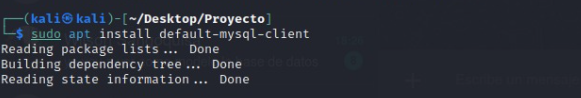
1. Se instala la última versión de Mysql-server y se descarga mysql

****

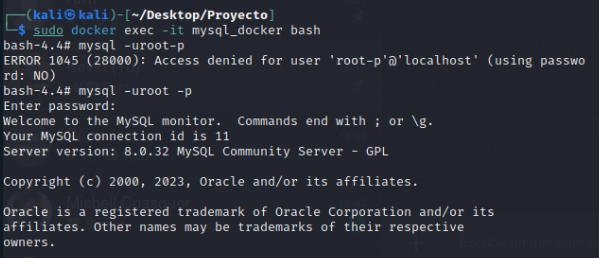
1. Se procede a ejecutar el contenedor usando la imagen instalada y se le nombra como mysql\_docker



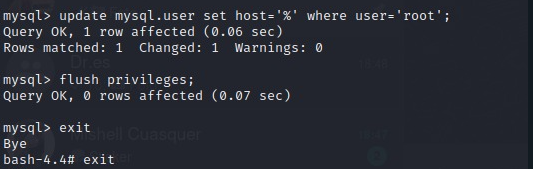
1. Se instala el servicio de mysql-client

****

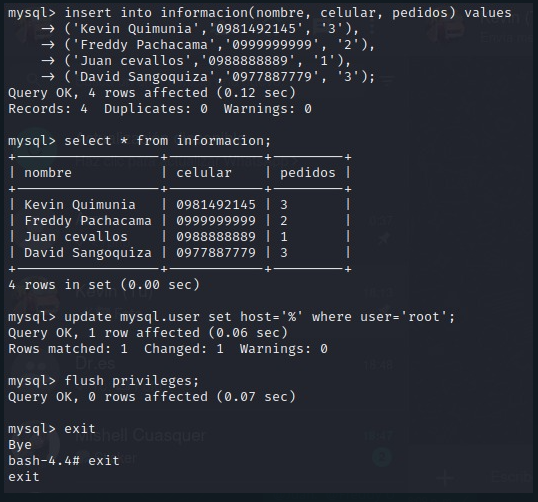
1. Se activa el bash del contenedor y se escribe el comando de acceso a mysql utilizando la contraseña del archivo logs



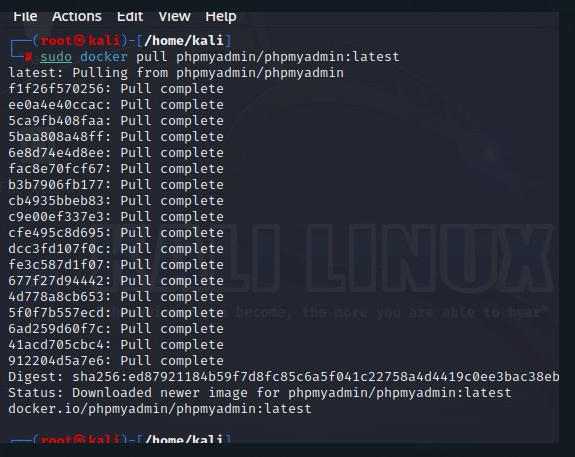
1. Se cambia la contraseña



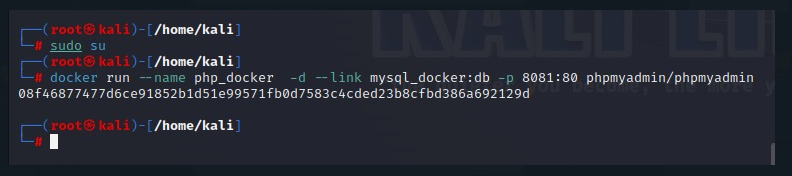
1. Se crea una base de datos para utilizarla como prueba y verificar que este en su funcionalidad total.



1. Se instala la imagen de phpMyAdmin en un nuevo contenedor

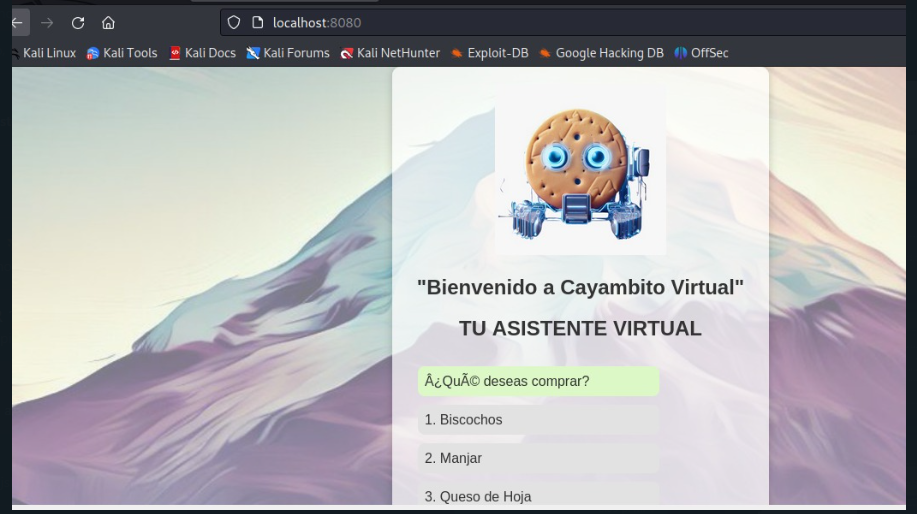


1. Se ejecuta el docker creado y se configura el puerto de enlace.

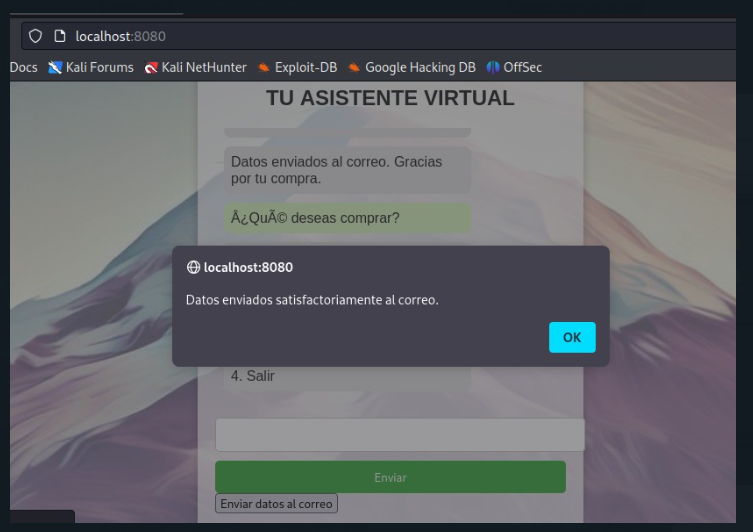


1. **Resultados obtenidos**

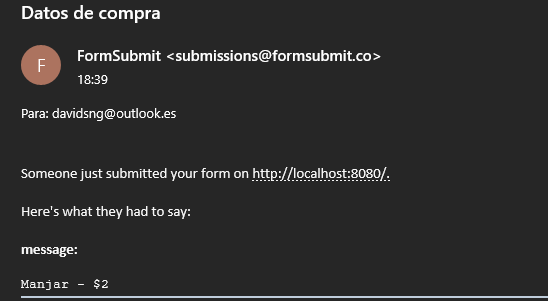
**Apache:** En el puerto 8080 podemos ver nuestro asistente virtual:

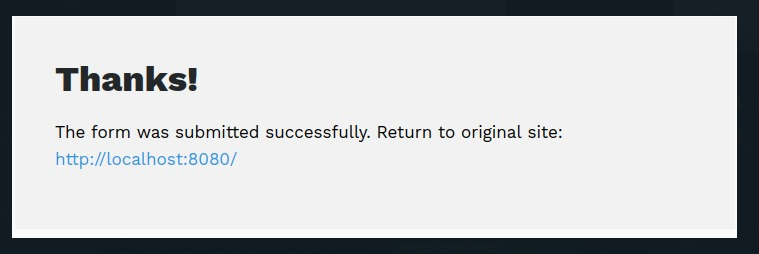


Tenemos tambien la función de que la compra realizada se envie al correo del cliente:

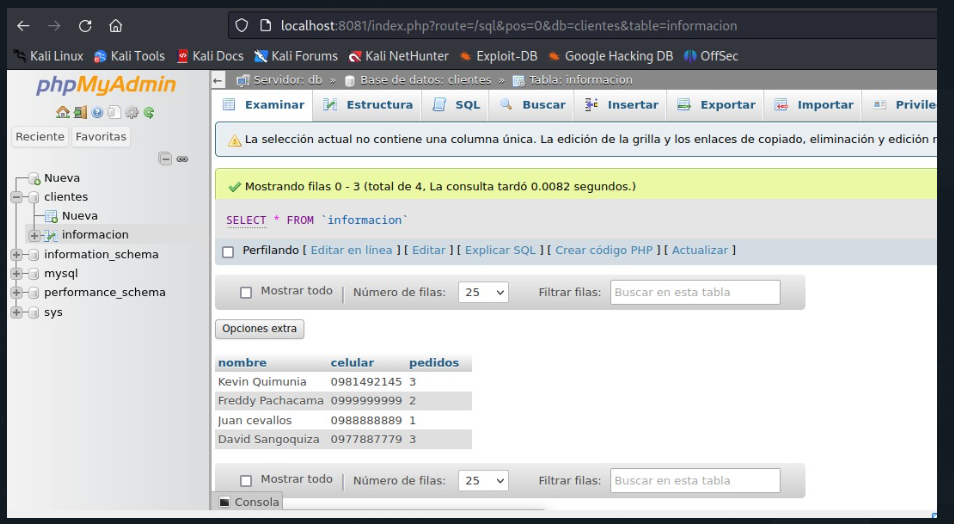


Correo recibido:

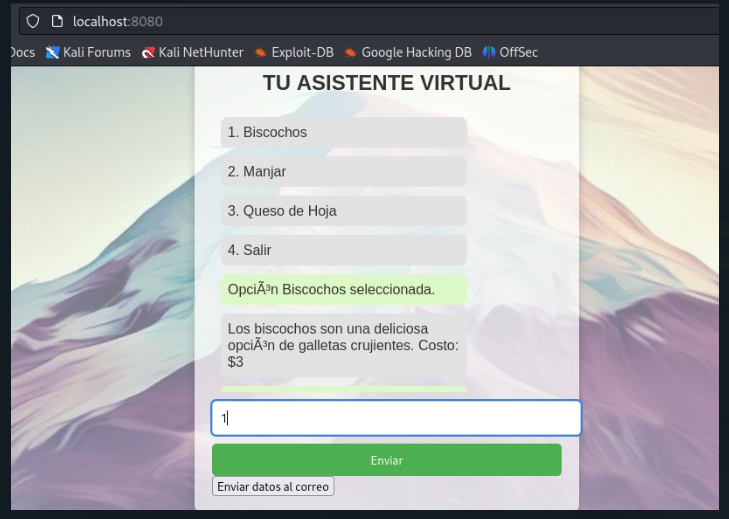




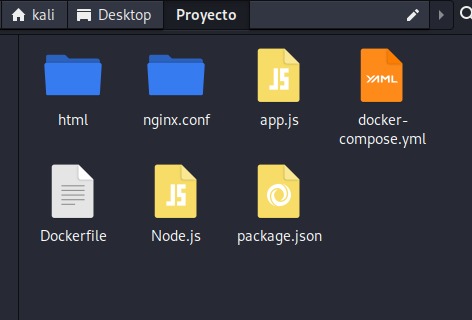
**Mysql:**

****

**Ngnix:**



**Contenedores:**



1. **Conclusiones**

* La implementación de microservicios en el proyecto del asistente virtual de una tienda online ha demostrado ser una elección acertada. Al dividir la aplicación monolítica en servicios independientes para la interacción del usuario (asistente virtual), la lógica empresarial (PHP) y el almacenamiento de datos (MySQL), se ha logrado una mayor flexibilidad y escalabilidad. Cada microservicio puede escalarse individualmente según la demanda, lo que permite manejar eficientemente picos de tráfico y mantener una experiencia de usuario fluida incluso en momentos de alta concurrencia.
* La adopción de microservicios ha promovido la modularidad y la independencia entre los distintos componentes del proyecto. Esto ha permitido que los equipos de desarrollo trabajen en paralelo, centrándose en áreas específicas de la aplicación sin afectar el funcionamiento de otros servicios.

1. **Recomendaciones**

* Es fundamental implementar un sólido sistema de monitoreo y recopilación de métricas para los microservicios y el asistente virtual. Utilizar herramientas como Prometheus, Grafana o similares permitirá tener una visión en tiempo real del rendimiento, uso de recursos y la salud de los servicios. Estos datos serán valiosos para detectar posibles cuellos de botella, anticipar problemas y realizar ajustes en la escalabilidad y optimización de los recursos.
* Dado que el proyecto involucra una tienda online, es vital priorizar la seguridad y protección de los datos del cliente. Implementar medidas de seguridad como autenticación, autorización y encriptación para proteger los datos sensibles, así como asegurarse de que las conexiones a la base de datos y los servicios estén debidamente protegidos. Además, realizar pruebas de seguridad periódicas y mantener las dependencias actualizadas ayudará a prevenir vulnerabilidades y proteger la integridad de la plataforma.

1. **Referencias bibliográficas**

* (S/f). Platzi.com. Recuperado el 21 de julio de 2023, de <https://platzi.com/tutoriales/1272-sql-mysql/5749-instalar-mysql-usando-docker-super-sencillo/>
* Cómo instalar Linux, Nginx, MySQL, PHP (LEMP) en Debian o Kali linux. (2014, enero 9). Mi vida Linux. <https://widrogo.wordpress.com/2014/01/09/como-instalar-linux-nginx-mysql-php-lemp-en-debian-o-kali-linux/>
* Marketing Centribal. (2022, enero 5). Aprende cómo crear tu asistente virtual. Centribal. <https://centribal.com/es/tu-asistente-virtual/>