Tabla de contenido

[Identificación y Definición de Microservicios 2](#_Toc167904011)

[Gestión de Usuarios (User Service): 2](#_Toc167904012)

[Catálogo de Productos (Product Service): 2](#_Toc167904013)

[Procesamiento de Pedidos (Order Service): 2](#_Toc167904014)

[Soporte al Cliente (Customer Support Service): 2](#_Toc167904015)

[Simulación de la Migración 3](#_Toc167904016)

[Roadmap de Migración 3](#_Toc167904017)

[Fase de Preparación: 3](#_Toc167904018)

[Fase de Migración Inicial: 3](#_Toc167904019)

[Migración del Servicio de Gestión de Usuarios: 3](#_Toc167904020)

[Migración del Servicio de Catálogo de Productos: 3](#_Toc167904021)

[Fase de Migración Intermedia: 3](#_Toc167904022)

[Migración del Servicio de Procesamiento de Pedidos: 3](#_Toc167904023)

[Migración del Servicio de Soporte al Cliente: 4](#_Toc167904024)

[Fase de Optimización y Monitoreo: 4](#_Toc167904025)

[Documentación de Arquitectura 4](#_Toc167904026)

[Microservicios y su Interacción: 4](#_Toc167904027)

[User Service: 4](#_Toc167904028)

[Product Service: 4](#_Toc167904029)

[Order Service: 4](#_Toc167904030)

[Customer Support Service: 5](#_Toc167904031)

[API Gateway: 5](#_Toc167904032)

[Redis: 5](#_Toc167904033)

[AWS EKS: 5](#_Toc167904034)

[Datadog: 5](#_Toc167904035)

[Diagrama de arquitectura 6](#_Toc167904036)

# Identificación y Definición de Microservicios

Basado en principios de diseño impulsado por el dominio (DDD), se han identificado los siguientes micro servicios y sus límites definidos:

## Gestión de Usuarios (User Service):

* **Responsabilidades**: Autenticación, autorización, manejo de roles y autorities y preferencias de usuario.
* **Comunicación**: Expone APIs REST para operaciones de usuario. Utiliza AWS Cognito para la gestión de sesiones.
* **Datos**: Almacena datos de usuario en MongoDB.

## Catálogo de Productos (Product Service):

* **Responsabilidades**: Gestión de productos, búsqueda y categorización de los mismos.
* **Comunicación**: Expone APIs REST para operaciones de productos.
* **Datos**: Almacena datos de productos en MongoDB.

## Procesamiento de Pedidos (Order Service):

* **Responsabilidades**: Gestión de carritos, manejo de pedidos, procesamiento de pagos, y seguimiento del historial de pedidos.
* **Comunicación**: Expone APIs REST para operaciones de pedidos.
* **Datos**: Almacena datos de pedidos en MongoDB.

## Soporte al Cliente (Customer Support Service):

* **Responsabilidades**: Manejo de tickets de soporte, devoluciones y aclaraciones.
* **Comunicación**: Expone APIs REST para operaciones de soporte al cliente.
* **Datos**: Almacena datos de tickets de soporte en MongoDB. Opcionalmente se puede integrar con Jira.

# Simulación de la Migración

## Roadmap de Migración

### Fase de Preparación:

* Evaluar el código monolítico actual y documentar las dependencias interna y externas.
* Configurar el entorno de microservicios utilizando AWS EKS para la orquestación de contenedores Docker. Para un control de cambios y gobierno de devops, se utilizará terraform para IaC.

### Fase de Migración Inicial:

### Migración del Servicio de Gestión de Usuarios:

* Desarrollar el microservicio de Gestión de Usuarios.
* Configurar la autenticación y autorización con AWS Cognito.
* Migrar datos de usuarios a MongoDB. Se evaluará usar AWS migration service.
* Desplegar y probar el servicio en AWS EKS, entorno DEV.

### Migración del Servicio de Catálogo de Productos:

* Desarrollar el microservicio de Catálogo de Productos.
* Migrar datos de productos a MongoDB.
* Desplegar y probar el servicio en AWS EKS, entorno DEV.

### Fase de Migración Intermedia:

### Migración del Servicio de Procesamiento de Pedidos:

* Desarrollar el microservicio de Procesamiento de Pedidos.
* Migrar datos de pedidos a MongoDB. Se evaluará usar AWS migration service.
* Configurar la comunicación con el microservicio de Gestión de Usuarios y Catálogo de Productos. Usando clientes rest y sqs.
* Desplegar y probar el servicio en AWS EKS, entorno DEV.

### Migración del Servicio de Soporte al Cliente:

* Desarrollar el microservicio de Soporte al Cliente.
* Migrar datos de soporte a MongoDB. Se evaluará usar AWS migration service.
* Configurar la comunicación con los otros microservicios relevantes.
* Desplegar y probar el servicio en AWS EKS. Entorno DEV.

### Fase de Optimización y Monitoreo:

* Implementar AWS Lambda para notificaciones vía email. Se conectará con SendGrid.
* Configurar Redis para cacheo.
* Configurar Datadog para monitoreo y logs.
* Monitorear el rendimiento y ajustar según sea necesario. CPU, memoría, número de replicas, entre otras configuraciones que permitan estabilizar los pods.

# Documentación de Arquitectura

## Microservicios y su Interacción:

### User Service:

* Proporciona APIs REST para operaciones de usuario.
* Utiliza AWS Cognito para la gestión de sesiones.
* Almacena datos en MongoDB.

### Product Service:

* Proporciona APIs REST para operaciones de productos.
* Almacena datos en MongoDB.

### Order Service:

* Proporciona APIs REST para operaciones de pedidos.
* Utiliza SQS para notificaciones asíncronas a otros servicios.
* Almacena datos en MongoDB.

### Customer Support Service:

* Proporciona APIs REST para operaciones de soporte.
* Almacena datos en MongoDB.

### API Gateway:

* Administra el enrutamiento y la autenticación de solicitudes hacía el sistema de e-commerce, ademas de proporcionar una capa de seguridad y sanitización de headers así como configurar cache y patrón de extrangulamiento.

### Redis:

* Utilizado para cachear respuestas y mejorar el rendimiento entre servicios y comunicación hacía base de datos.

### AWS EKS:

* Orquesta y gestiona contenedores Docker para los microservicios.

### Datadog:

* Monitorea y gestiona logs de los microservicios.

**Descomposición por Dominio**: Los microservicios fueron definidos basados en las áreas funcionales clave de la aplicación monolítica. Esto permite una gestión más clara y la escalabilidad independiente de cada componente.

**Comunicación Asíncrona**: La elección de AWS SQS para comunicación asíncrona permite una mayor tolerancia a fallos y desacoplamiento entre servicios. Adicional que nos permite configurar dead letter para el reprocesamiento de pedidos.

**Cacheo con Redis**: La utilización de Redis para cachear consultas frecuentes reduce la carga en la base de datos y mejora la velocidad de respuesta.

**Autenticación Centralizada con AWS Cognito**: Facilita la gestión de usuarios y mejora la seguridad al delegar la autenticación a un servicio especializado.

**Orquestación con AWS EKS**: Permite el despliegue y la gestión eficiente de contenedores Docker, asegurando escalabilidad y alta disponibilidad.

**Monitoreo con Datadog**: Proporciona visibilidad integral de los logs y el rendimiento de los microservicios, facilitando la detección y resolución de problemas.

## Diagrama de arquitectura

