**Creación de una plataforma de comercio electrónico.**

Este documento tiene como finalidad describir las funcionalidades técnicas de la plataforma de comercio electrónico, además de facilitar la realización de mejoras y mantenimiento de la misma.

**Objetivos:**

El objetivo de esta plataforma es brindarle al usuario final una facilidad de compra de productos que resuelvan sus necesidades de forma segura, eficiente y confiable.

**Alcance:**

Desarrollar una plataforma de comercio electrónico que funcione a base de micro servicios, capaz de ser escalable y se pueda desplegar en un ambiente dockerizado, del lado de la seguridad se va a manejar un gateway de Spring Cloud con un servidor de configuración.

(Checar si es viable meter Jenkins o alguna herramienta parecida).

Por el momento se queda fuera el sistema de pagos.

**Interesados:**

* Carlos Armando Calderón Espinosa
* Erick Montalvo
* Christhian Lugo Govea
* Diego

**Hitos y cronograma**

\*Inserte calendario de fechas\*

**Requisitos funcionales:**

* El sistema deberá permitir a un usuario seleccionar una cantidad definida de productos, ingresarlos a un carrito de compras y facilitar posteriormente la compra segura de los mismos.

**Roles de usuario y permisos:**

Los roles y funciones de los usuario son:

* Administrador - Dar de alta usuarios, productos y órdenes de compra.
* Cliente - Buscar productos, agregarlos al carrito y generar órdenes de compra.
* Vendedor - Listar productos y su stock.

**Carácteristicas:**

1. Servicio de registro de usuarios
2. Servicio que autentique los usuarios
3. Servicio de catálogos (Productos y Categorías del producto).
4. Servicio que gestione el carrito de compras.
5. Servicio que genere la orden de compra con base en la gestión de pedidos(Checkout)
6. Servicio que procese el pago en línea y se integre a una pasarela de pagos.
7. Servicio que proporcione reseñas y calificaciones de los productos.

**Requisitos no funcionales:**

El sistema tiene como objetivo alcanzar un nivel óptimo en rendimiento ya que se van a utilizar tecnologías flexibles y compatibles para el manejo de paginación de productos.

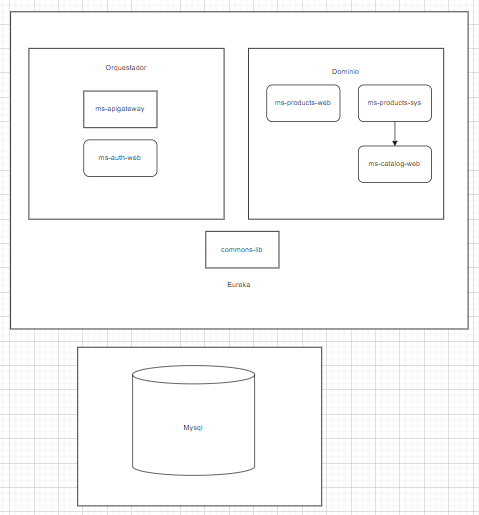
El sistema se podrá escalar en ambos ejes debido a que se está usando una arquitectura orientada a microservicios y esta nos permite desacoplar componentes y posicionarnos en equipos diferentes.

En cuanto a la seguridad, las peticiones al sistema serán orquestadas mediante un Gateway de Spring Cloud el cual será el responsable de evaluar si el usuario logueado aun cuenta con un token vigente para realizar la operación solicitada. También los datos que vayan transitando de servicio a servició y los que sean sensibles que estén almacenados en la base de datos serán cifrados para tener una mayor robustez en la seguridad.

Para la confiabilidad se estará monitoreando la carga de actividad de los servicios mediante herramientas como Prometheus o Grafana. Con esto podremos detectar fallos y reportarlos al área correspondiente para su resolución en un futuro despliegue.

En cuestión de la mantenibilidad se pretende hacer un pipeline que permita a los desarrolladores desplegar una versión de QA para probar las nuevas funcionalidades y posteriormente ya que los cambios sean aprobados pasar a una versión productiva.

**Diagrama de Arquitectura de alto nivel:**



**Componentes:**

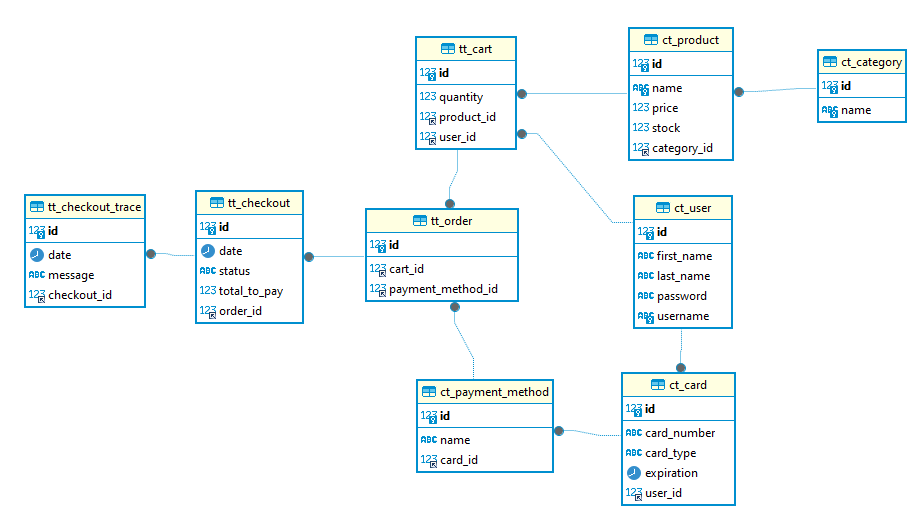
* Como frontend se desarrollará un sitio web desarrollado en Angular con Bootstrap 5, de backend se desarrollará en Spring boot con apoyo de los módulos de Spring Security, Spring Cloud Configuration Server, Spring Cloud Gateway.
* Se está decidiendo usar una base de datos relacional en MySQL debido a que el diseño de la aplicación; todos los componentes tienen relación entre sí. Además de que para los interesados es más fácil de implementar y dockerizar.
* Para la parte de la autenticación se piensa usar la librería de Jwtt con un cifrado BCrypt (Posteriormente se implementará un servicio como KeyCloak)
* Para el almacenamiento de imágenes de los productos se va a usar un bucket en S3.
* Para la integración y despliegue continuo se va a utilizar Github actions.

**Motor de base de datos:**

Se selecciona una base de datos relacional debido a que las entidades participantes cuentan con relaciones definidas y claras entre sí, además de que es importante mantener integridad entre las transacciones que se desea almacenar.

**Esquema de base de datos:**

\* Inserte DER sin relaciones\*

**Diagrama de Relación de Entidades (ERD):**  


**Desarrollo:**

Se utilizarán los siguientes frameworks y tecnologías para el desarrollo de la aplicación:

* Java 17
* Spring Framework
* Spring Boot
* Spring Security
* Hibernate
* MySQL
* Angular

**Versionamiento del código:**

A continuación el link a los repositorios GIT de código:

<https://github.com/orgs/RunningWithSissors/teams/metaphorce-sala-1/repositories>

**Implementación de microservicios y APIs:**

\* Inserte LINK a documentación de swagger\*

**Medidas de seguridad:**

Spring Cloud Gateway se encargará de validar, autorizar e invalidar los tokens que se vayan recibiendo en las peticiones de los clientes que estén tratando de accesar al sistema, también será el responsable de proteger los endpoints de los servicios para autorizar sólo a los roles que tengan los permisos especificados en el sistema. Se cifrará toda la información para el envío y recepción de la misma por parte del frontend cliente.

Para el monitoreo y auditoría de los servicios se utilizará AWS CloudWatch, el cual registrará las actividades que se están presentando desde que el sistema recibe una solicitud hasta que la termina.

**Pruebas y validaciones:**

Consideramos importante alcanzar al menos un 100% de resultado exitoso (80% de cobertura) sobre las pruebas integrales que se realicen sobre el sistema.

* Pruebas unitarias
* Pruebas de integración

**Despliegue y mantenimiento:**

Se considera importante la implementación de un pipeline el cual se encargue de integrar los distintos cambios del equipo y realizar la liberación continua de cambios a los ambientes que correspondan.

Se considera implementar las siguientes tecnologías:

* Jenkins
* Docker
* Kubernetes

**Plan de Mantenimiento:**

\* Estrategias de monitoreo y mantenimiento implementadas \*

**Estrategias de Escalabilidad:**

\* Planes para escalar el sistema conforme aumente la demanda \*