**Análisis Prueba Técnica**

1. **Preguntas para realizar:**

* ¿En caso de seleccionar más productos que no estén en stock que debería hacer la aplicación?
* ¿Se puede realizar un rollback de una orden de pago, en los escenarios en que un cliente desee cancelar la compra?
* ¿Cuántas ordenes de compra se pueden generar por usuario en una misma fecha?
* ¿En un futuro se piensa agregar más medios de pago y abarcar más países que lo estipulado en el requerimiento? Esto con el fin de que la construcción del desarrollo sea genérica y este abierta a incorporar estos cambios a futuro de manera fácil.

1. **Diseño aplicación:**

**Diagrama

Descripción generada automáticamente**

En lo anterior se refleja un modelo MER a alto nivel del flujo del proceso del requerimiento de usuario. Se contempla la realización de un microservicio con el patrón de diseño MVC “Modelo-Vista-Controlador” que contenga la lógica de la solución. Este desarrollo contara con lo siguiente:

* Sonar: Para la verificación del código estático para identificar los puntos susceptibles de mejora.
* Pruebas de seguridad SCA: Para verificar que el código estático no cuente con vulnerabilidades de seguridad ej: “SQLINJECTION, SCP, CORS”
* Selenium: Se desarrollará la respectiva automatización de pruebas del componente.
* PU: Se realiza pruebas unitarias utilizando las tecnologías Junit y **Mockito.**
* DockerFile: Para el despliegue en ambiente cloud del componente
* Shell con la automatización del proceso de despliegue para integración con Jenkins o gitLab
* Utilización log4j para logs
* Jacoco: para la verificación de la cobertura de las pruebas

**NOTA:** La aplicación de prueba cuenta con lo necesaria para funcionar y cumplir con lo requerido. No se contempló lo siguiente “Selenium, jacoco, pruebas SCA”. Pero en un escenario real se tiene que contemplar.

**Plus**: Se pueden generar N Ordenes de pago para un mismo cliente o diferentes clientes Se utilizaron las siguientes tecnologías “Docker, Junit, Mockito, SonarQube”.

**Resultado pruebas microservicio:**

Compilación exitosa junto con los 2 test exitosos de la aplicación, el primero crea la orden de la compra y el segundo la consulta:

Texto

Descripción generada automáticamente

Se levanta contenedor:

Captura de pantalla de computadora

Descripción generada automáticamente

Prueba desde Postman: Se pueden enviar varias peticiones y se van almacenando en un MAP simulando una BD real:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Se envía otra generación de orden de pago para el mismo cliente con la misma fecha de entregaInterfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

Se envía otra generación de orden de pago con diferente fecha

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Consulta por fecha, retorna todas las ordenes de pago por fecha de entrega

http://127.0.0.1:8080/b2chat/consultaOrdenCompra/2022-07-02

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Con la otra fecha

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico, Teams

Descripción generada automáticamente

La solución de Código se sube a GitHub:

https://github.com/jonatan182/b2chatPrueba