Concepto de Rendimiento para Ezytech V2 (con pago por app): Insatisfactorio

El rendimiento durante la prueba de carga ***es insatisfactorio***, como lo evidencian los bajos valores de APDEX en la mayoría de las transacciones críticas, la alta incidencia de errores 4XX y los problemas de código subyacentes como NullPointerException.

Es crucial llevar a cabo una revisión exhaustiva del código y la infraestructura para identificar y resolver estos problemas antes de considerar la aplicación como estable y confiable**.** ***Además se recomienda no implementar el pago manual desde la app en producción*** hasta que estos errores hayan sido solucionados.

- **Rendimiento Insatisfactorio (Rojo)**: La mayoría de las transacciones, como "registrarSalidaPagoAutomatico", "registrarSalidaSinEntrada", "pagar", y "registrarSalidaPagoManual", tienen un APDEX en rojo. Esto sugiere que los usuarios experimentan tiempos de respuesta significativamente lentos, lo cual es inaceptable en un entorno de producción. Estas transacciones son críticas porque están directamente relacionadas con el procesamiento de pagos y salidas, lo que podría afectar gravemente la experiencia del usuario y, en última instancia, la confianza en el sistema.
- **Rendimiento Condicionado (Amarillo)**: "consultarServiciosEnCurso" tiene un APDEX en amarillo, lo que indica que el rendimiento está en una zona de advertencia. Aunque no es tan insatisfactorio como las transacciones en rojo, aún existe un riesgo significativo de que los usuarios perciban un rendimiento inadecuado.
- **Rendimiento Satisfactorio (Verde)**: Solo la transacción "consultarPlaca" tiene un APDEX satisfactorio, lo que significa que los usuarios encuentran este servicio confiable y rápido. Sin embargo, este es un caso aislado, y no es suficiente para compensar las otras áreas problemáticas.

Errores Durante la Prueba:

La tabla de errores proporciona información adicional sobre los problemas específicos que afectan el rendimiento:

- **Errores 500 (Errores del Servidor)**: su aparición en los logs del sistema podría ser indicativa de ***problemas más profundos en la infraestructura del servidor, como sobrecarga o problemas con la base de datos***. Un error 500 podría causar una cascada de errores 4XX, especialmente si el backend no puede manejar las solicitudes correctamente y falla en autenticar o autorizar a los usuarios.
- **Errores 4XX (401 y 403)**: Estos códigos de error indican problemas de autenticación (401) y permisos (403). La alta frecuencia de estos errores sugiere que el *sistema* ***no está manejando correctamente las solicitudes de los usuarios***, posiblemente debido a configuraciones incorrectas en los permisos o problemas con la sesión de usuario durante la carga. Estos errores pueden afectar gravemente la capacidad de los usuarios para completar transacciones.

- **java.lang.NullPointerException*: Este error es crítico y sugiere que hay fallos en el manejo de datos dentro del código de la aplicación***. Específicamente, este tipo de error ocurre cuando el sistema intenta operar con un objeto que es null, lo que indica falta de validación o manejo de excepciones. Dado que este error está relacionado con el procesamiento de pagos, podría estar causando fallos en el procesamiento de transacciones, lo que contribuye al mal desempeño general de las funcionalidades relacionadas con el pago.

Relación entre Errores 4XX y 500:

Los errores 4XX, especialmente 401 y 403, suelen ser problemas de cliente relacionados con autenticación y autorización, pero pueden ser inducidos por errores 500 en el servidor. Si el servidor tiene ***un fallo crítico (500)*** que afecta su capacidad de manejar sesiones o consultas a la base de datos, esto podría resultar en que las solicitudes legítimas del cliente no puedan ser autenticadas o autorizadas, desencadenando errores 4XX.

Por ejemplo:

- Si un servidor sufre un error 500 durante una autenticación, la sesión del usuario podría no ser creada correctamente, lo que llevaría a un error 401 cuando el usuario intente realizar otra acción.
- De manera similar, un error 500 en la gestión de permisos podría causar que el servidor no reconozca los permisos del usuario, resultando en un error 403.

Recomendaciones:

1. Manejo Adecuado de Errores en el Código:

- Manejo de Excepciones: Desarrollar un manejo robusto de excepciones para capturar y manejar errores inesperados sin que estos provoquen fallos en cascada. Utilizar bloques try-catch en las áreas críticas del código y asegúrar de que todas las excepciones sean registradas y gestionadas adecuadamente.
- Mensajes de Error Informativos: Los mensajes de error deben ser claros y útiles tanto para los usuarios como para los desarrolladores. Implementar mensajes de error personalizados que proporcionen información específica sobre lo que salió mal.

2. Mejoras en la Autenticación y Autorización:

- Revisión de Sesiones y Tokens: Revisar cómo se gestionan las sesiones y los tokens de autenticación. Asegúrar de que no hayan expiraciones inesperadas o problemas de sincronización que puedan causar errores 401.
- Políticas de Autorización: Verificar que las políticas de autorización estén correctamente implementadas y que todos los permisos necesarios estén configurados adecuadamente.

4. Pruebas y Monitoreo:

- Pruebas de Carga Continuas: Realiza pruebas de carga regulares para identificar problemas antes de que lleguen a producción. Estas pruebas deberían simular escenarios reales y extremos para evaluar la robustez del sistema.
- Monitoreo en Tiempo Real: Implementar herramientas de monitoreo en tiempo real para observar el rendimiento y detectar problemas antes de que afecten a los usuarios.
- Alertas Proactivas: Configura alertas proactivas que se disparen cuando ciertos umbrales críticos se alcanzan (por ejemplo, tiempo de respuesta elevado, tasa de error alta, etc.). Esto permitirá al equipo de operaciones intervenir antes de que un problema afecte a los usuarios finales.

5. Optimización del Código:

- Profiling del Código: Realiza un profiling del código para identificar funciones que consumen mucho tiempo o recursos. Herramientas como VisualVM o JProfiler pueden ayudar a identificar cuellos de botella en el rendimiento del código.6. Mejora en la Gestión de Dependencias: