SISTEMAS DISTRIBUIDOS

PROYECTO FINAL

Simulación de un Sistema Financiero de Dinero Electrónico

Elaborado por: M. en C. Ukranio Coronilla

Este proyecto será programado en equipos de tres alumnos que serán asignados de entre mis tres grupos de Sistemas Distribuidos y cuya lista será publicada en teams. Por su complejidad equivale a dos proyectos individuales, en este caso la calificación obtenida se asigna al proyecto 5 y 6.

En este proyecto se va a simular un sistema donde los usuarios van a depositar dinero desde su cuenta bancaria para convertirlo en dinero electrónico con el cual pueden hacer transferencias, pagos o recibir dinero de otros usuarios dentro de la misma plataforma. Un ejemplo similar a este sería el de Mercado Pago.

Debido a que se hace una manipulación del dinero de los usuarios, la aplicación debe ser segura, con un buen desempeño y tolerante a fallos. Para cumplir con estos requisitos se va a desarrollar e implementar como un sistema distribuido, utilizando el lenguaje de programación Java con arquitectura de microservicios REST, desplegada en Google Cloud Platform y/o alguna otra plataforma Cloud junto con servicios en la nube.

Requisitos indispensables del proyecto (si no cumple con alguno de ellos no será evaluado su proyecto y su calificación será de cero):

- Todo el código backend del proyecto debe estar en lenguaje Java (no es válido el uso de Pyton u otro lenguaje de programación). Sólo se permite el código necesario para incorporar el frontend en otros lenguajes.
- Los códigos de servidores y clientes HTTP deben reutilizar los códigos proporcionados en clase.

Requisitos del proyecto

1. Debe basarse en una arquitectura de microservicios que se comunican mediante REST.
2. Debe implementar un servicio de autenticación para los usuarios **AuthService** que reciba las credenciales del usuario (CURP y contraseña), las valide y devuelva un token JWT(JSON Web Tokens) firmado digitalmente. Ese token se usa luego en cada solicitud a otros servicios, sin necesidad de volver a enviar las credenciales.
3. Debe implementar un servicio de cuentas **AccountService** el cual le permite a una cuenta del usuario obtener su balance, hacer depósitos (de la cuenta del banco al monedero

electrónico), hacer retiros (del monedero electrónico al banco) o hacer transferencias de dinero electrónico entre los usuarios.

4. Toda la información del sistema además de saldos y transacciones debe ser almacenada en Google Cloud Storage, y/o en una base de datos en la nube.
5. Debe implementar un servicio de transacciones **TransactionService**, el cual recibe eventos de transacciones desde una cola Pub/Sub. Este servicio consume el evento, realiza la transferencia, actualiza balances y hace una publicación cuando la transacción esté confirmada.
6. Debe implementar un Servicio de Auditoría **AuditService** que escucha también el tópico Pub/Sub y guarda registros de cada transacción.
7. Ambos servicios **AccountService** y **TransactionService** deben estar replicados para mejorar la disponibilidad y el rendimiento.
8. Debe existir un programa en Java que se ejecuta en una consola y que simula n clientes interactuando en el sistema con ayuda de h hilos donde ($1 < h < 9$). Al inicio todos los clientes depositan una cantidad p de pesos y realiza cada uno aproximadamente t transacciones por minuto (con montos aleatorios y en intervalos aleatorios de tiempo) donde ($1 < t < 60$) sin terminar nunca. Dicho programa recibe como parámetros n, h, p y t.
9. Debe contar con una interfaz que utilice la librería de Java Lanterna, encargada de monitorear en intervalos de tiempo de n segundos (n se pasa como parámetro del programa) el uso de CPU de todas las instancias de procesamiento que conformen su sistema, con el nombre del servicio que está corriendo y con la IP de la instancia donde se ejecuta.
10. Debe contar con una interfaz web para los usuarios que les permite darse de alta como clientes del sistema, hacer depósitos, retiros y transferencias. También le permite ver al usuario su monto total y un log con timestamps de todas las transacciones que se han realizado en su cuenta.
11. Debe contar con una interfaz web para el administrador, que le permita observar los balances de cada usuario, un gráfico con el número de transacciones por periodo de tiempo, un gráfico con el monto de las transacciones por periodo de tiempo, logs con timestamps de todas las transacciones que ha realizado cada usuario y el monto total de dinero de todos los usuarios del sistema.

Pruebas a las que se someterá el sistema

1. En la interfaz web del usuario crear cuenta, hacer depósitos, transferencias, retiros y comprobar que los saldos se actualizan correctamente en la interfaz web del administrador.
2. Se ejecuta el programa que simula n clientes realizando transacciones en el sistema, verificando que no existan cambios en el saldo total y se mantenga la consistencia al repetir simultáneamente la prueba 1.
3. Simular la caída de uno de los servicios **TransactionService** para verificar que sigue funcionando y posteriormente simular la caída del segundo servicio mientras hay mensajes en Pub/Sub para verificar que, al reiniciarse, un servicio procesa correctamente los mensajes pendientes y al reiniciar el otro servicio sigue procesando correctamente.
4. Simular la caída de uno de los servicios **AccountService** para verificar que sigue funcionando y posteriormente simular la caída del segundo servicio mientras hay

mensajes en Pub/Sub para verificar que, al reiniciarse, un servicio procesa correctamente los mensajes pendientes y al reiniciar el otro servicio sigue procesando correctamente.

Concurso

En el momento de iniciar la evaluación todos los coordinadores de cada equipo tendrán que enviarme por mensaje privado las URLs de las páginas web del usuario y del administrador. Las cuales haré públicas para que puedan acceder simultáneamente todos los alumnos (en teoría los tres grupos de alumnos) y comprobar su funcionamiento realizando las pruebas que consideren necesarias. En caso de que detecten algún error podrán enviarme el mensaje correspondiente por teams reportando el tipo de error para tenerlo en consideración en la calificación del proyecto.

Así mismo se realizará una votación por todos los alumnos para decidir cuál fue la interfaz web del usuario y del administrador que brindaron la mejor experiencia en su uso. A los miembros del equipo ganador se les otorgarán dos puntos en la **calificación final del curso**, un punto en la **calificación final del curso** al equipo que quede en segundo lugar y medio punto en la **calificación final del curso** al equipo que quede en tercer lugar.

Importante: Todos los puntos del proyecto serán revisados detalladamente con un checklist, por lo que para obtener 10 se requiere cumplir con todos ellos.

La revisión del proyecto final (explicación del código y funcionamiento) se realizará únicamente el lunes 12 de enero comenzando a las 8:00 AM y hasta que termine la evaluación de todos los equipos, lo cual les haré saber con un mensaje grupal en la plataforma de teams. Durante este tiempo el sitio web de todos los equipos debe encontrarse en funcionamiento para que cualquier alumno del curso pueda acceder al mismo.

Cada equipo deberá entregar todo el código del proyecto final con la consideración de que, si se detecta un plagio, ambos equipos (plagiado y plagiario) tendrán una calificación de cero.

Si un alumno no trabajó o hubo quien trabajó menos que los demás tendrán que hacérmelo saber en el momento de la revisión para ponderar la calificación de los miembros del equipo.