Parcial 2 Software

1. Banco

    a. Los atributos de calidad que se deberían tener en cuenta para esta solución son:

        - Seguridad: ya que al tratarse de un banco y manejar algo tan delicado como el dinero, se debe tener en cuenta los temas de privacidad.
        - Performance: ya que se requiere que debe tener tiempos de respuesta inferiores a 200 milisegundos.
        - Extensibilidad: ya que debe ser posible agregar nuevas funcionalidades.
        - Resiliencia: debe tener la capacidad de recuperarse rápido de problemas al tratarse de un banco.

    b. Los patrones arquitectónicos que consideraríamos implementar para el banco sería:
        - Microservicios orquestación: pues este patrón arquitectónico se debe tener en cuenta al implementar operaciones ACID, tal como lo que se aplica en un banco, ya que si por ejemplo en alguna parte no se completa cierta funcionalidad, no puede quedar perdido el dinero, sino que lo ideal sería cancelarlo todo, es decir, en este caso nos sirve un orquestador para controlar la secuencia de servicios que deben operar.
        - También se podría combinar con API Gateway por temas de seguridad y escalabilidad.
        - Así mismo una Access token debería ser necesaria para no ingresar directamente a mi cuenta por ejemplo, sino tener que estar autenticado primero, y que por temas de seguridad se cierre la cesión después de cierto tiempo.

2. Para microservicios aplicados con coreografía, podríamos hablar de un escenario de una tienda virtual ya que los servicios podrían coordinarse entre sí de una manera no centralizada, entonces digamos uno se encarga de realizar la compra, otro de actualizar el inventario, otro de entregar la notificación. Por otro lado, para microservicios aplicados con orquestación, podríamos imaginar un escenario de una aplicación de viajes donde se deba reservar tanto el vuelo como la reserva del hotel y la renta de un automóvil. En este caso, se requeriría una coordinación centralizada para asegurar que todos los servicios se coordinen correctamente. Por ejemplo, si un cliente reserva un vuelo, un hotel y un automóvil, es necesario asegurarse de que las fechas y horas de las reservas no se superpongan y que todo esté correctamente reservado antes de confirmar la reserva. Así como mencionamos anteriormente, con orquestación se utilizan para transacciones ACID en todo el flujo de trabajo para garantizar que si algo sale mal en cualquier parte del proceso, se pueda revertir la reserva en su totalidad.

3. Escenario Event-based pattern:
   Un escenario en donde se vería aplicado el patrón de event-based pattern sería por ejemplo, uno de una transportadora, que tiene cientos de camiones para transporte de envíos en sudamérica, estos camiones deben enviar su localización cada 5 minutos, y debe haber una página donde las personas puedan ver la localización de su pedido, es decir donde va, por lo tanto estas localizaciones deben estar mandándose y almacenándose en una base de datos, y como son tantas se gestionan con ayuda de servicios y colas para poder mandar a la base de datos, y que las personas puedan visualizar la locación. Así que también si hay alta demanda para ver las localizaciones (alta cantidad de usuarios), sus peticiones también se pueden ir gestionando con ayuda de las colas y microservicios.
4. El requerimiento que incluye explícitamente un requerimiento no funcional es b ya que menciona que el usuario debidamente autenticado podrá efectuar transferencias a terceros del mismo banco, ya que al decir debidamente autenticado, alude a un tema de seguridad.
5. La afirmación es falsa ya que la integración contínua se refiere a una práctica que consiste en hacer integraciones automáticas de un proyecto lo más a menudo posible para así poder detectar fallos cuanto antes, lo que se menciona en la aplicación se refiere a la entrega continua.