Respuestas

1. Problemas de diseño

Para aceptar más transacciones adaptándonos al cuello de botella de la base de datos sin hacer vertical o horizontal scaling y mejorar el sistema durante períodos de alta demanda se puede implementar un sistema de cola (queueing) para las transacciones y connection pooling.

El sistema de cola (queueing) se puede implementar con tecnologías como Redis o RabbitMQ y esto permite al sistema aceptar transacciones solicitadas cuando la base de datos está en máxima capacidad e ir procesando las transacciones a medida que la base de datos libera capacidad sin sobrecargarse. El sistema de cola sigue el orden de First In, First Out (FIFO) que significa que la primera transacción solicitada se procesa primero.

Cada vez que se solicita una transacción a una base de datos SQL ocurre un proceso de múltiples pasos:

1. La aplicación usa un driver para abrir una conexión.
2. Se abre un network socket para conectar la aplicación y la base de datos.
3. La operación se completa y la conexión puede que se cierre.
4. EL network socket se cierra.

Durante baja demanda estos pasos no incurren demasiado poder de procesamiento por parte del sistema pero durante alta demanda abrir y cerrar conexiones frecuentemente si requieren de mucho procesamiento. Para resolver esto se puede hacer connection pooling que consiste en mantener la conexión a la base de datos abierta, pasándole operación a operación como sea necesario y así evitar abrir y cerrar conexiones que afectan la performance del sistema.

---

4. Aprendizaje

Client Apps:

- El proyecto cuenta con dos frontend para los usuarios. Uno es una página web y el otro es una aplicación de móvil. Ambas se comunican con el backend a través de un API Gateway.

API Gateway:

- Acepta solicitudes que vienen de los frontend, las procesa y las dirige a los microservicios que corresponden.

Microservices:

- Es un estilo de arquitectura para desarollar aplicaciones que permiten dividir aplicaciones entre partes independientes y en este caso cada microservicio cuenta con su propia base de datos.
  Los microservicios de Catalog y Shopping Cart contactan el message broker.

Message Broker:

- Esta parte no estoy seguro, pero me imagino que recibe por un lado información sobre el catálogo y aplicá los descuentos correspondientes y por el otro recibe los carritos de los usuarios y crea las órdenes de compra usando un queueing message system.

---

5. Demostrando tus hallazgos

Para explicar esta nueva tecnología y sus beneficios a los stakeholders se puede utilizar una analogía sobre cómo funciona una fábrica.

Imaginemos que la fábrica tiene solo una línea de producción para todas las órdenes. En esta línea de producción pasan todos los procesos desde gestionar el stock disponible hasta ejecutar el pago de la órden de compra. Esto está bien hasta que algo se rompe en la línea de producción evitando la entrada de nuevas órdenes ó la demanda de órdenes se incrementa, éstas se demoran y el cliente tiene una mala experiencia.

Ahora, pensemos en que la línea de producción se subdivide en partes independientes que trabajan de manera coordinada, una por ejemplo gestiona el inventario, otra chequea la información del usuario y otra procesa el pago. Esto vendrían a ser los microservicios. Ahora, el tema de escabilidad. Los proveedores de microservicios permiten escalar de manera rápida y eficiente de acuerdo a la demanda, como por ejemplo refiriéndonos de vuelta a la analogía de la fábrica, si se necesitan más líneas de producción por la demanda se crean nuevas líneas de producción y si la demanda es baja se pueden reducir las líneas de producción operando de manera eficiente y manteniendo los costos de acuerdo a la demanda real.

Esto permite que las órdenes se procesen rápido, que el usuario tenga una experiencia fluída, positiva y satisfactoria que puede lograr que vuelva como cliente o recomendar la buena experiencia a otras personas.