1. Enunciado (Proyecto final - Curso de Computación en internet)

Tu equipo ha sido contratado para desarrollar una tienda en línea para una empresa local. La tienda necesita tener la funcionalidad básica de permitir a los administradores agregar productos y a los clientes comprarlos. Además, se debe generar una factura para cada compra realizada por un cliente. La gestión de datos debe tener algún tipo de persistencia, puede ser con el manejo de archivos tipo JSON (no es necesario, pero también podría usar algún motor de base de datos), además es necesario que integre funcionalidades de autenticación (puede ser con JWT).

Requisitos Funcionales:

1. Roles de Usuario:

- La tienda tendrá dos roles de usuario: Administrador y Cliente.
- El administrador tiene permiso para agregar nuevos productos al inventario de la tienda.
- El cliente puede navegar por los productos disponibles, agregar productos al carrito y realizar una compra.

2. Servicios Prestados:

Administrador:

- Iniciar sesión en su cuenta de administrador.
- Agregar nuevos productos al inventario de la tienda, especificando el nombre del producto, descripción, precio y cantidad disponible.

Cliente:

- Registrarse o iniciar sesión en su cuenta de cliente.
- Ver la lista de productos disponibles en la tienda.
- Agregar productos al carrito de compras.
- Realizar una compra, generando una factura que incluya los detalles de los productos comprados, la cantidad y el precio total.
- Ver el historial de compras anteriores.

2. Análisis de Requisitos del Proyecto y Justificación de la Elección de Bases de Datos

En el desarrollo de la tienda en línea, hemos decidido usar dos bases de datos: MongoDB como base de datos NoSQL para el carrito de compras y PostgreSQL como base de datos relacional para gestionar usuarios e historial de compras.

2.1. Uso de PostgreSQL (Relacional) para Usuarios e Historial de Compras

Gestión de Usuarios:

Los usuarios de la tienda tienen roles definidos (Administrador y Cliente) y deben realizar varias operaciones que requieren integridad de datos y relaciones claras, tales como el inicio de sesión, el registro, y la visualización de su historial de compras. Para gestionar esta información de manera eficiente, PostgreSQL es ideal, ya que nos permite organizar

la información de los usuarios en tablas estructuradas, aprovechando su capacidad de manejar relaciones entre entidades.

En PostgreSQL, creamos la tabla de usuarios con las siguientes columnas relevantes:

- user_id: Identificador único del usuario.
- username: Nombre de usuario único.
- user_password: Contraseña encriptada del usuario.
- user_role: Rol asignado (Administrador o Cliente).

Esta estructura relacional facilita la creación de índices que optimizan las consultas de búsqueda (por ejemplo, búsquedas rápidas por correo electrónico o nombre de usuario) y garantiza la integridad referencial entre los usuarios y su historial de compras.

Gestión del Historial de Compras:

El historial de compras de los usuarios debe incluir detalles como los productos comprados, el total de la compra y la fecha de la transacción. Dado que esta información es estructurada y necesita ser consultada de manera eficiente, utilizar una base de datos relacional como PostgreSQL es lo más adecuado. Las compras estarán asociadas a los usuarios mediante una relación, permitiendo un fácil acceso al historial de compras de cada usuario.

En este caso, almacenamos el historial de compras en una tabla con las siguientes columnas:

- id: Identificador único de la compra.
- user_id: Referencia al ID del usuario que realizó la compra.
- products: Una lista de productos comprados, que puede ser almacenada como un objeto JSONB.
- total: Total de la compra.
- purchase_date: Fecha de la compra.

La relación entre los usuarios y su historial de compras se establece mediante una clave foránea (user_id), lo que asegura la integridad de los datos y permite realizar consultas complejas (como obtener todas las compras de un usuario, o consultar compras realizadas en un rango de fechas).

2. 2. Uso de MongoDB (NoSQL) para el Carrito de Compras

MongoDB se utiliza para gestionar el carrito de compras de los usuarios debido a su flexibilidad y eficiencia en la manipulación de datos no estructurados o semi-estructurados. A continuación, se justifican los motivos por los cuales MongoDB es la elección adecuada para esta parte del sistema:

Escalabilidad horizontal: MongoDB es una base de datos diseñada para ser escalable horizontalmente, lo que significa que puede manejar grandes cantidades de datos distribuidos entre varios servidores. En un sistema de comercio electrónico, donde se pueden generar múltiples carritos de compras y cada carrito puede crecer rápidamente en términos de productos, la escalabilidad de MongoDB garantiza que el sistema pueda manejar un aumento en la carga de trabajo sin degradar el rendimiento.

3. Alta disponibilidad: MongoDB ofrece opciones de replicación y alta disponibilidad a través de su Replica Set. Esto es útil en escenarios donde es crítico tener acceso constante a los datos, como en el proceso de compra en línea. **Si un servidor se cae, otro servidor replica automáticamente los datos para garantizar que los carritos de compras estén disponibles en todo momento.**

4. Documentos embebidos para eficiencia: En nuestro modelo, los productos se almacenan como un arreglo dentro del documento del carrito. Esto es ventajoso en MongoDB porque permite que todos los datos relacionados con el carrito de compras (productos, cantidad, precios) se almacenen en un solo documento. **Esto hace que las lecturas sean más rápidas, ya que no necesitas hacer varias consultas o uniones de tablas para obtener todos los productos en el carrito.**

5. Facilidad de mantenimiento: El hecho de que MongoDB sea un sistema schema-less (sin esquema predefinido) facilita el mantenimiento de la base de datos cuando se deben realizar cambios. Por ejemplo, si más adelante se desea agregar más atributos a los productos (como una categoría, una fecha de caducidad o un código de descuento), se puede hacer fácilmente sin afectar otras colecciones o registros en la base de datos.

6. Compatibilidad con JSON para integración de APIs: MongoDB usa JSON (BSON) para almacenar los datos, lo que facilita la integración con otras partes de la aplicación, especialmente cuando se manejan datos a través de APIs que también usan este formato. Al estar alineado con las tecnologías modernas (como Node.js o frameworks basados en JavaScript), MongoDB facilita el trabajo con datos estructurados de manera similar en ambas direcciones: cliente-servidor.

3. Comparación con Otras Soluciones NoSQL (como Cassandra)

1. Consultas ad-hoc: MongoDB permite consultas dinámicas y flexibles, esenciales para manejar los carritos de compras, mientras que Cassandra está optimizado para consultas predefinidas y no es tan flexible.

2. Escalabilidad y consistencia: MongoDB ofrece un buen equilibrio entre escalabilidad y consistencia, lo que es crucial para asegurar que los datos del carrito sean correctos. Cassandra, por su parte, prioriza la disponibilidad sobre la consistencia, lo cual podría ser problemático en este contexto.

3. Facilidad de uso y administración: MongoDB tiene una curva de aprendizaje más baja y herramientas como MongoDB Atlas y Compass que facilitan la administración, monitoreo y escalabilidad. Cassandra, al ser más compleja en su configuración y mantenimiento, requiere conocimientos más profundos en términos de ajuste de rendimiento y gestión de clústeres.

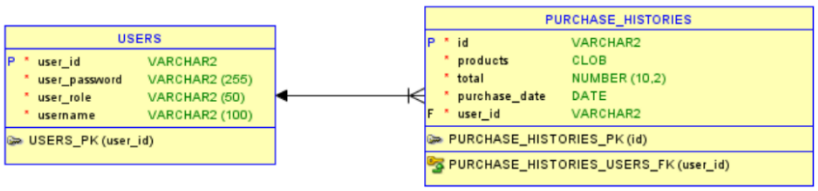
4. Modelado de datos más intuitivo: En MongoDB, los carritos de compras se pueden modelar de manera sencilla y natural utilizando documentos anidados. Esto se ajusta bien a la estructura de datos flexible y dinámica del carrito, mientras que Cassandra requiere una mayor planificación y optimización para manejar relaciones entre datos de manera eficiente, lo que complica el desarrollo del sistema.

4. Conclusión

La elección de PostgreSQL y MongoDB para la tienda en línea se basa en las fortalezas de cada base de datos para necesidades específicas. PostgreSQL se utiliza para gestionar usuarios e historial de compras debido a su capacidad de manejar datos estructurados, relaciones y consultas complejas con integridad referencial. En cambio, MongoDB se emplea para el carrito de compras, aprovechando su flexibilidad, escalabilidad horizontal y alta disponibilidad, lo que lo hace ideal para manejar grandes volúmenes de datos no estructurados de manera eficiente. Comparado con soluciones como Cassandra, MongoDB ofrece un mejor equilibrio entre escalabilidad y consistencia, y una administración más sencilla. En conjunto, ambas bases de datos optimizan el rendimiento y la fiabilidad del sistema.

5. Modelos de datos

Relacional:



No relacional: