Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bases de Datos II

Proyecto I: Cartzo

Integrantes:
Jason Barrantes Arce (2015048456)
Steven Bonilla Zúñiga (2015056296)
Jorge González Rodríguez (2015083567)

Profesor: Erick Hernández Bonilla

9 de abril, 2017
San José, Costa Rica

Introducción

El presente proyecto consiste acerca de un sistema de compra de artículos en línea, talvez realizando la comparación adecuada del sistema con el sistema de Amazon en pequeña escala, el sistema debe cumplir las necesidades básicas de un sistema de compras como el de Amazon, tales como registrar usuarios, mantener un inventario de productos con fotos, agregar productos a un carrito en línea y realizar las compras. Todas estas funciones son básicas de un sistema de compra en línea que debe realizar, en nuestro actual sistema de compra en línea, muchas de estas opciones estarán disponibles para todos los clientes.

Entre las funciones básicas a cumplir o que cumple el proyecto son las siguientes:

- Se crean los usuarios que van a usar la aplicación, distinguiendo un menú para administradores y otro para clientes.
- Los usuarios son creados y almacenados en la base Neo4j.
- Los administradores pueden agregar artículos y guardar su respectiva imagen. Todas las imágenes van a ser almacenadas en MongoDB.
- Los usuarios pueden almacenar los productos que desean comprar previamente en un carrito, este carrito es editable y es almacenado temporalmente en redis.
- Los usuarios pueden realizar un comentario y una evaluación del producto si lo desean. La calificación será de 1 a 5. Se puede realizar un comentario antes de comprar el producto, pero en la calificación solo se realizará después de haber comprado el producto. Ambos son almacenados en Cassandra.
- Si los usuarios gustan pueden seguir las compras de otros usuarios.
- Los usuarios pueden ver las opciones que han comprado.

Presentación y Análisis del problema

En esta sección se analizarán los problemas encontrados en el programa y su respectiva solución, además de un análisis de la estructuradas utilizadas:

Diseño utilizado:

- Se utilizaron cinco bases de datos distintas para guardar toda la información que se va a usar en el programa:

- Las imágenes se almacenan en MongoDB debido a que son archivos difíciles de guardar en bases relacionales, esta fue la razón que nos motivó a usar una base NoSQL.
- Para guardar las relaciones se decidió utilizar Neo4j debido a que es una base de datos de grafos, y eso nos facilita relacionar los datos de las bases para seguir un usuario.
- Para guardar los comentarios y las calificaciones se utilizó Cassandra, esto principalmente porque se tomó en cuenta que un usuario puede realizar un comentario y una puntuación sobre un producto, esta fue la razón para poder almacenar la información mediante columnas como lo hace Cassandra.
- Para guardar los productos antes de verificar la compra, es necesario el uso de un carrito, donde se almacenan temporalmente los productos. Este carrito será almacenado en Redis para poder acceder de forma rápida y adrede.
- Las órdenes son los productos del carrito que se decidieron comprar, estas se van a almacenar en MySQL, junto al inventario y el producto, esto se almacenará en MySQL debido a que es información que está relacionada entre sí.

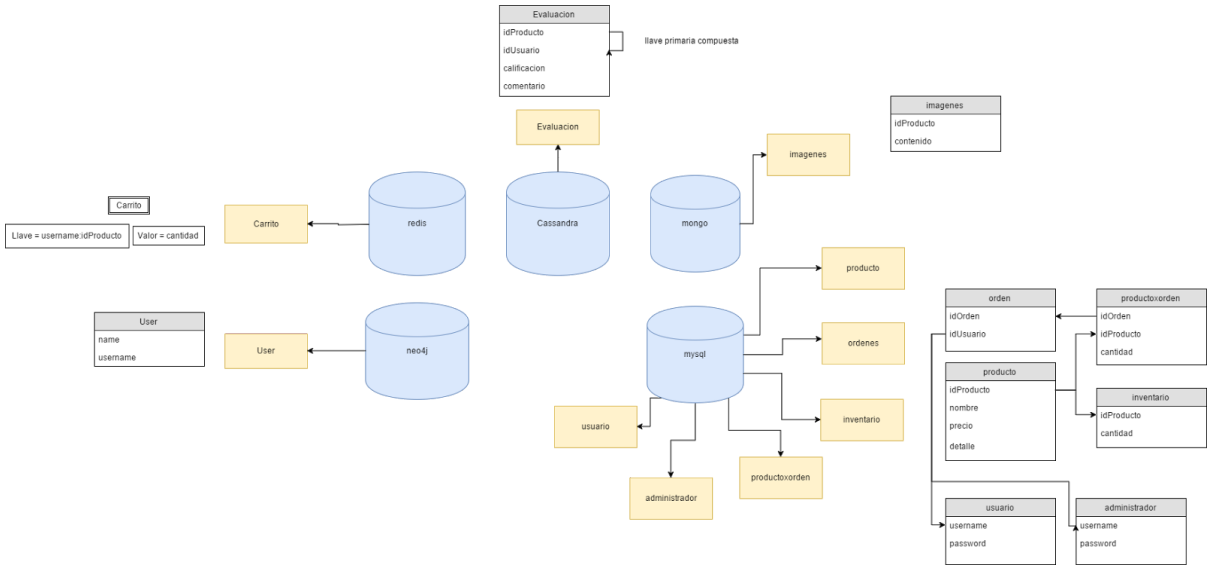
Aclaraciones con base en el diseño :

- Las calificaciones son almacenadas en cassandra debido a que se considera que es la mejor forma de acceder a estos valores, en Cassandra se encuentra compuesta por una tabla junto al comentario.
- El inventario es almacenado en MySQL, puede ser que se deba un poco por la mentalidad Relacional obtenida en el curso de Bases de datos I, y el ejemplo más claro para nosotros es que un inventario sea una tabla de relaciones, por esta razón se decidió implementar en MySQL.
- Se consideró que las recomendaciones no eran necesario de guardar en ninguna tabla, debido a que pueden cambiar dependiendo de la venta del producto o de la calificación obtenida.

MVC

Para el proyecto se utilizó .NET Framework 4.5 de Visual Studio utilizando el modelo de MVC. Se dividió el modelo en Views, Controllers, Models y con estos se desarrollaron las partes de código. Las comunicaciones entre las capas corresponde de la siguiente manera, en la capa modelo se ubica la información o datos con los cuales se van a operar, se encarga también de gestionar la misma, ya sea actualizar o consultar, además de realizar las conexiones con las bases de datos; luego la capa controladora se encarga de responder a los eventos que realiza el usuario en la capa de vista y de realizar peticiones de datos a la capa modelo, por lo tanto el controlador es un mediador entre la vista y el modelo; por último se encuentra la capa vista que es la parte visual del programa, el usuario puede interactuar con la interfaz y es capaz de realizar peticiones en la misma.

Diagrama de las bases de datos



El siguiente diagrama muestra la arquitectura utilizada para la base de datos.

En la base neo4j se encuentran los nodos de la clase User los cuales contienen el name y username; en caso de Redis se utiliza como llave el username con el idProducto debido a que están asociados con el carrito, mientras que en valor se contiene la cantidad que desea comprar el usuario; seguidamente tenemos en Cassandra una tabla Evaluación, la cual contiene el idProducto, el idUsuario, calificación y comentario; para el caso de MongoDB se almacena el idProducto y el contenido de imagen; por último en MySQL se usan las tablas orden, producto, productoxorden, inventario, usuario y administrador; cada uno relacionado con sus tablas respectivas.

Conclusiones

- El uso de MVC favorece el manejo de las interfaces mediante el grupo de clases Controllers, lo que es sumamente recomendable cuando se ocupa tener una división clara de las interfaces con la lógica del programa. En cambio con Models, se puede llegar a qué Models es utilizada para realizar operaciones atómicas o comunicaciones directamente con la base de datos.

- El lenguaje de DML y DDL que maneja Cassandra es bastante fácil de adaptar en comparación al de los otros lenguajes de las bases de datos, esto porque Cassandra es parcial a SQL y esto facilita el conocimiento que se tiene previamente de las bases relacionales. Aún con este detalle, hay ciertas diferencias que se pueden considerar ineficientes en el llamado de la base de datos por medio de C#, podría decirse que realiza un tipo de SQL injection para llamar a la base de datos, lo que nos parece sumamente peligroso.

- En MongoDB, decidimos almacenar las imágenes en Buckets, en especial los buckets otorgados por FSGrid que posee dos colecciones llamadas fs.files y fs.chunks, pero el fs.files es solo el acceso al fs.chunks que posee el contenido lógico. Podría decirse que el fs.files es sólo el acceso con la información del almacenamiento y el fs.chunks es el contenido físico de la imagen. Esta metodología de almacenamiento es bastante eficiente en comparación con el almacenamiento regular de una colección por aparte, el problema está en que se utiliza más espacio por cada una de las imágenes.

Recomendaciones

- El uso de otras plataformas o de otros métodos de diseño como el modelo de negocios Controladora, Interfaz, Métodos que son relativamente parecidos al MVC pudo haber sido implementado en el programa.

- Respetar las divisiones respectivas hechas que realiza el MVC, que es más eficiente para ambiente de trabajo como por ejemplo Visual Studio.

- Investigar de diversos procesos para evitar SQL Injection en la conexión con Cassandra, serían buenas estrategias para defenderse en caso de Hackers.

- El almacenamiento de las imágenes en MongoDB mediante un GridFS es más sencillo de implementar pero puede ser sumamente costoso, debido a que GridFS posee un almacenamiento extra innecesario y puede almacenar hasta 16 MB por archivo, mucho de este espacio puede llegar a llenarse sin ser utilizado. Considerando la anterior situación, se debería reconsiderar otro tipo de estrategia para almacenar en Mongo o encontrar una forma de reducir el espacio por defecto que utiliza Mongo en sus chunks.

Bibliografía

Anderson R. (2017). Getting started with ASP.NET Core MVC and Visual Studio. marzo 22, 2017, de Microsoft Sitio web: https://docs.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc

Hernández U. MVC (Model, View, Controller). marzo 22, 2017, de codigoFacilito Sitio web: https://codigofacilito.com/articulos/mvc-model-view-controller-explicado

RobinHoody. (2016). cypher examples. marzo 23, 2017, de Github Sitio web: https://github.com/Readify/Neo4jClient/wiki/cypher-examples

Ojeda I.. (2015). Using REDIS Cache with C#. marzo 22, 2017, de C#Corner Sitio web: http://www.c-sharpcorner.com/UploadFile/2cc834/using-redis-cache-with-C-Sharp/

Schumacher R.. (2012). Getting Started with Apache Cassandra on Windows the Easy Way. marzo 22, 2017, de Datastax Sitio web: https://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way

Rivkin S.. (2016). How to MongoDB in C# - Part 1. marzo 22, 2017, de OzCode Sitio web: https://blog.oz-code.com/how-to-mongodb-in-c-part-1/