Ecommerce SQL y NoSQL

**Profesor/a**

* Godio, Claudio José

**Alumnos:**

* Von Elm, Lucas – LU: 1133639
* Alippi, Juan Ignacio – LU: 1139842
* Muttolo, Joaquín – LU: 1146061
* Mendieta, Juan Ignacio – LU: 1135071

Tabla de Contenidos

[Introducción 2](#_Toc139444557)

[Bases de datos utilizadas 2](#_Toc139444558)

[MongoDB 2](#_Toc139444559)

[Catálogo de Producto 2](#_Toc139444560)

[Listado de Precios 3](#_Toc139444561)

[Usuarios 3](#_Toc139444562)

[Listado de Pedidos 3](#_Toc139444563)

[Redis 4](#_Toc139444564)

[Cassandra 4](#_Toc139444565)

[SQL 4](#_Toc139444566)

[Conclusión 6](#_Toc139444567)

# Introducción

Este proyecto consiste en el desarrollo de una aplicación que haga uso de varias bases de datos distintas para las funciones de almacenar usuarios, productos, facturas y otras relacionadas.

Hicimos uso de varias bases de datos tanto SQL como NoSQL para las funcionalidades donde mejor convenía, demostrando la funcionalidad de cada una y su importancia en el almacenamiento de datos.

# Bases de datos utilizadas

## MongoDB

Aprovechamos el sistema de documentos y autogeneración de id única de mongo para el almacenamiento de datos que no tengan una estructura fija con relaciones y que no requieran de las propiedades ACID.

La importancia de que no existan una estructura fija es el mayor punto a favor de utilizar esta base de datos para estos casos. Estos son datos que se basan en guardar la mayor cantidad de información posible, y eso significa que en un futuro sea necesario agregar más campos en cada documento de los que había en un inicio. Esto sería una tarea muy cara de realizar si usáramos SQL.

Además Mongo es muy útil al momento de guardar datos y que no requieran de ninguna relación con otros documentos.

Las colecciones utilizadas son detalladas a continuación:

### Catálogo de Producto

Nombre de colección en MongoDB: CatalogoProductos

Aquí almacenamos en documentos los productos que se ingresan en el sistema, junto a sus datos varios. El precio de los productos es almacenado en una colección distinta (Listado de Precios). Decidimos almacenar la información en Mongo debido a que resulta beneficioso no contar con una estructura definida, sino que podemos agregar o quitar campos según sea necesario para el caso.

### Listado de Precios

Nombre de colección en MongoDB: ListadoPrecios

Aquí almacenamos en documentos el listado de los precios de los productos. Decidimos hacerlo en una colección distinta por varias razones:

* Eficacia al realizar búsquedas: Ciertas funcionalidades del sistema requieren el precio del producto y nada más, al buscar estos datos en una colección con solo los precios ahorramos en búsquedas más complejas en documentos ultra extensos con todos los otros datos del producto mismo.
* Expansible: Si deseamos guardar descuentos, bonificaciones, cupones u otras cosas que modifiquen al precio, podemos almacenar esto con el precio. Con esto evitamos que el documento en el Catálogo de Productos sea demasiado grande.

### Usuarios

Nombre de colección en MongoDB: Usuarios

En esta colección almacenamos la información de los usuarios y sus sesiones dentro del sistema. Los usuarios son considerados clientes dentro del proyecto. En Mongo almacenamos toda la información que tenemos del usuario y que es relevante para el sistema.

Aunque también almacenamos información del usuario en SQL, esta es solo para la parte de facturas con la información relevante solo para ese punto. En MongoDB nos encargamos de almacenar el completo de la información que se dispone del cliente.

### Listado de Pedidos

Nombre de colección en MongoDB: ListadoPedidos

En esta colección guardamos un registro de los pedidos realizados en el sistema, con la información del carro de compra, el cliente y el operador responsable. Los pedidos son documentos que pueden cambiar en la cantidad de campos que contienen, así que por ello decidimos utilizar MongoDB para almacenarlos.

## Redis

Se utiliza Redis para el Carro de Compra. Esto es por la gran utilidad de la escritura rápida de la base de datos y el uso de Hashs con clave valor.

Estructura de los hash en redis: “ key – field – value “

La forma en la que utilizamos es la siguiente:

1. Creamos un Hash con la “key” del carro de compras. Para esta “key” utilizamos la idUsuario que nos genera Mongo, asegurándonos de que sea única y que sea fácil de relacionar a un usuario con su carro de compras.
2. Dentro utilizamos como “field” a la id de producto que nos genera mongo al guardarlo. Así podemos saber de qué producto se está hablando.
3. Como “value” utilizamos la cantidad de ese producto que está en el carro de compras.

Así es como podemos obtener de un carro de compras específico todos los productos que se agregó a este y la cantidad de estos.

## Cassandra

En Cassandra almacenamos toda la información de loggeo sobre cambios realizados en un producto, como cambio de descripción, precio, especificaciones, etc.

La información del loggeo lo realizamos en una tabla donde almacenamos una copia de la información del producto en su estado actual.

La Row Key de la tabla está conformada por:

Partition Key: IdProducto – Para poder filtrar por productos

Clustering Key: Timestamp – Un registro de la fecha de realización del cambio.

## SQL

Utilizamos SQL para el almacenamiento de las facturas realizadas, ya que consideramos que es información esencial para el negocio y que cumpla con las propiedades ACID.

Si hubiéramos decidido utilizar bases de datos NoSQL no podríamos garantizar en todo momento que estén disponibles, o que los cambios que se realicen no queden completos a media, ni tampoco que sean consistentes entre todos los clústers. Esto podría generar conflictos entre la información que se ingresó y la que se encuentra almacenada, y con un tema muy sensible como es la facturación haría imposible que este negocio fuera sostenible.

La información que decidimos almacenar en la base de datos es:

* Información del cliente
* Información del operador responsable del proceso
* Información del pago

En otra tabla de facturas almacenamos una referencia a cada una de las filas de la siguiente manera:

A screenshot of a computer

Description automatically generated

La estructura de la tabla esta verificada que cumpla con las tres primeras reglas de normalización.

# Conclusión

El Desarrollo del Proyecto demuestra como cada producto de Bases de Datos tiene una utilidad única para aportar a cada problema que se tiene. Como desarrolladores de Software nuestro trabajo es diseñar y emplear soluciones a los problemas que se nos dan, y el poder emplear diversos métodos de almacenamiento de datos para cada necesidad es una habilidad esencial.

En la actualidad, la necesidad de almacenar la mayor cantidad de datos es una cuestión presente en todo negocio. Mientras que en décadas pasadas se realizaba las bases de datos SQL se presentaban como la única opción, debido a la tecnología del momento y diversos factores, creemos que esto va a cambiar. En vez de analizar solamente cuanto espacio necesito para guardar la información, es importante ver el problema desde otro punto de vista, analizando donde conviene mejor almacenarla.