Ecommerce DBII

Profesor:

* Godio, Claudio José

Alumnos:

* Von Elm, Lucas
* Alippi, Juan Ignacio
* Muttolo, Joaquín
* Mendieta, Juan Ignacio

Table of Contents

No table of contents entries found.

# 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)

### 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

### Listado de Pedidos

Nombre de colección en MongoDB: ListadoPedidos

## 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. Asi podemos saber de que producto se esta hablando.
3. Como “value” utilizamos la cantidad de ese producto que esta en el carro de compras.

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

Un carro de compras es algo que no requiere de varios datos, solo saber que producto y cantidad se pide, básicamente, clave valor, y justamente Redis es una base de datos que utiliza Clave-Valor que resulta muy útil por esto.

Cassandra

SQL