Para este sistema decidimos usar MongoDB para modelar la mayoría de los datos: los clientes y sus teléfonos, las facturas y los detalles, y el nombre, marca y descripción de los productos (la información que cambiaría menos seguido). Y usamos Redis para el stock y precio de los productos.

Alguna de las razones por las que decidimos usar Mongo son:

- Ofrecer consistencia en la información, para asegurarse que cuando un cliente actualiza su dirección o le llegue una factura vea la información con la certeza de que es correcta.
- Las queries requerían un gran número de argumentos compuestos entre diferentes tablas, lo cual se puede concretar de forma sencilla con la sintaxis de MongoDB. Se decidió modelar la información en 2 tablas, una para los clientes y otra para las facturas, los teléfonos como un array dentro de los clientes y los detalles de factura como un array dentro de los documentos de facturas.

Se eligió utilizar Redis porque:

- Ofrece consistencia y velocidad en la información, que resulta ser crítico al momento de mantener actualizado el stock y precio de los productos, y así evitar problemas como vender algo de lo cual se agotó el stock.
- Consideramos que solo la información de precio y stock era necesaria en acceso rápido dado que el nombre y descripción de un producto no suele ser modificada.

Analizamos otras alternativas de bases NoSQL para los distintos datos a almacenar, y descartamos varias de ellas.

- Cassandra para los clientes y teléfonos: originalmente consideramos esta opción, ya que resultaba eficiente para la búsqueda de clientes por nombre y teléfonos por ID de cliente, pudiéndose definir estos como partition key para la búsqueda (o creando índices para estos). Sin embargo, muchas de las consultas requerían devolver todos los datos de la base, sin buscar por ningún atributo en específico (u obtener los clientes que no tuvieran factura, por ejemplo, lo cual requería lo mismo), para lo cual Cassandra no resultaba una muy buena opción.
- Consideramos después tener únicamente los teléfonos en Cassandra, con los clientes en MongoDB, lo cual permitía una buena búsqueda por número de cliente (definiendo esta como PK), pero nuevamente se pedía la información de todos los teléfonos, y también incluirlos cuando se mostraban los datos de todos los clientes, y en ningún caso según los patrones de acceso resultaba mejor esto que tener los teléfonos directamente guardados con los clientes (siempre que se pide la información de estos es en conjunto con la del cliente).
- Cassandra para los productos: parecía útil para la búsqueda por marca o por ID de los productos, pero nuevamente se complicaba al tener que mostrar los productos que no habían sido facturados, y además consideramos que para los datos de precio y stock era de gran importancia la consistencia (aunque en Cassandra se pueda regular el nivel de consistencia, su intención es la disponibilidad, por lo que consideramos que era mejor otra alternativa).
- Neo4j para las relaciones entre las entidades (sus IDs) y MongoDB para guardar información adicional: agregaba complejidad innecesaria e ineficiencia a las queries, resultando en la necesidad de saltar más de dos veces entre ambas bases de datos para varias de las queries. En términos generales, descartamos Neo4j porque el sistema solicitado no parecía centrarse en relaciones entre entidades, éstas no tenían más de un nivel de profundidad (es más una relación directa entre cliente y teléfonos, factura con cliente o detalle, detalle con producto).
- Redis para clientes y/o teléfonos: lo consideramos brevemente, pero requería demasiadas búsquedas por valores que no eran una clave, lo cual complicaba la estructura de querer hacerlo en Redis, y en ciertos casos resultaba en tener que realizar múltiples consultas de más para obtener los datos requeridos.