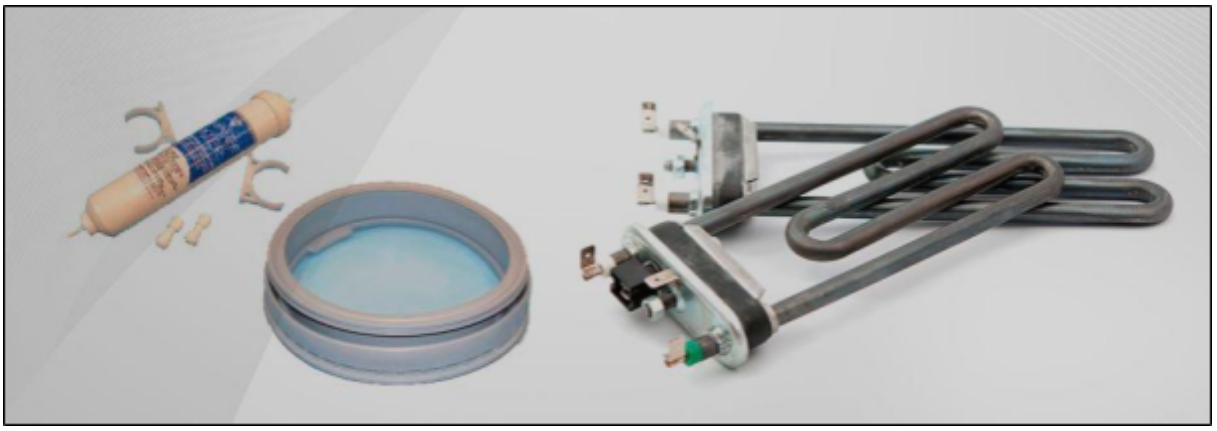
GENERALITAT VALENCIANA

Carlos Collado Calvo
DAM 1°

Microproyecto:

Inventario de piezas de electrodomésticos

Versión MongoDB



Carlos Collado Calvo

24/05/2024

ÍNDICE

1. INTRODUCCIÓN

El presente proyecto se centra en el desarrollo y la implementación de una base de datos de inventario utilizando MongoDB, una de las bases de datos NoSQL más populares y versátiles del mercado. En un inicio este mismo inventario se desarrolló utilizando MySQL, sin embargo en un entorno empresarial moderno, la gestión eficiente del inventario es crucial para mantener operaciones fluidas y garantizar que los productos estén disponibles cuando se necesiten y MongoDB puede proporcionar una solución más robusta y escalable para el manejo de datos de inventario.

En este trabajo se abordarán varios aspectos clave:

- **Herramientas y métodos utilizados**: Se explicarán las herramientas y métodos de los cuales se harán uso y, sobre todo, las razones por las que MongoDB puede ser una opción más práctica que MySQL, destacando sus beneficios.

- **Perspectiva dinámica:** Al haber partido de la base de un proyecto MySQL, en este apartado se describirá, paso a paso, el diseño de toda la base de datos haciendo una comparativa entre como se hizo en su día para MySQL y cómo se realizará para MongoDB.

    - ➔ **Diseño de la base de datos**: Se detallará la estructura de la base de datos, incluyendo las colecciones y los documentos que almacenarán la información relevante del inventario, como productos, categorías, proveedores y transacciones.

    - ➔ **Paso a tablas**: Se mostrará la implementación del patrón de diseño que hace uso de múltiples colecciones para gestionar el almacenamiento de datos de manera eficiente y estructurada.

    - ➔ **CRUD**: Se describirá el proceso de configuración e implementación de la base de datos, incluyendo la inserción de datos, la creación de índices para optimizar las consultas y la integración con aplicaciones de software que permitan la visualización y manipulación de los datos del inventario.

- **Perspectiva dinámica**: Se presentarán ejemplos prácticos de cómo la base de datos puede ser utilizada para realizar un seguimiento del inventario, generar reportes y facilitar la toma de decisiones. También se incluirán consultas típicas que se pueden ejecutar para obtener información valiosa, como el estado del stock, el historial de transacciones y la identificación de tendencias de consumo.

- **Conclusiones**: Finalmente, se ofrecerá un análisis de los resultados obtenidos, destacando las mejoras en la gestión del inventario logradas gracias a la implementación de MongoDB, así como posibles mejoras futuras y ampliaciones del sistema.

Este proyecto pretende demostrar no solo la viabilidad técnica de utilizar MongoDB para la gestión de inventarios, sino también los beneficios prácticos que esta tecnología puede aportar a una organización. La elección de MongoDB permite manejar datos complejos y en constante cambio de manera eficiente, ofreciendo una base sólida para el crecimiento y la adaptación a las necesidades empresariales.

2. HERRAMIENTAS Y MÉTODOS

En el desarrollo de este proyecto se han utilizado diversas herramientas y métodos que han facilitado tanto el diseño como la implementación del sistema. A continuación, se ofrece una breve descripción de cada una de las herramientas usadas:

- **MongoDB**: Base de datos NoSQL orientada a documentos que permite el almacenamiento y la gestión de datos en un formato flexible y escalable. Su capacidad para manejar grandes volúmenes de datos y su modelo de datos basado en documentos JSON la hacen ideal para aplicaciones que requieren rapidez y eficiencia en la gestión de información.

- **Visual Studio Code** (VSCode): Editor de código fuente altamente personalizable y extensible, desarrollado por Microsoft. Se ha utilizado para la escritura y edición de scripts de base de datos, así como para el desarrollo de aplicaciones auxiliares. Su integración con diversas extensiones para MongoDB permite una experiencia de desarrollo más fluida y productiva. Al ser el IDE que he ido usando durante el curso además, ya tenía cierta fluidez con su manejo.

- **Draw.IO**: Herramienta de diagramación en línea que permite crear diagramas de flujo, diagramas de entidad-relación (ER) y otros tipos de diagramas. En este proyecto, se ha utilizado para diseñar la estructura de la base de datos y visualizar las relaciones entre diferentes colecciones y documentos.

- **Dia**: Aplicación de diagramación de código abierto similar a Draw.IO. Se ha utilizado como una alternativa para la creación de diagramas ER y otros esquemas necesarios para planificar la base de datos. Su interfaz sencilla y su capacidad para exportar diagramas en varios formatos lo hacen una herramienta útil para el diseño de sistemas.

- **ChatGPT**: Modelo de lenguaje desarrollado por OpenAI, que ha sido utilizado como una herramienta de apoyo durante el desarrollo del proyecto. A través de ChatGPT, se ha obtenido asistencia en la resolución de dudas técnicas y generación de ideas para optimizar el diseño y la implementación del sistema de inventario.

- **Apuntes**: Los apuntes tomados durante el curso y a lo largo del proceso de desarrollo han sido una fuente fundamental de referencia. Estos apuntes incluyen conceptos teóricos, mejores prácticas y ejemplos prácticos que han guiado las decisiones de diseño y han proporcionado una base para la implementación del proyecto.

Cada una de estas herramientas ha desempeñado un importante papel en el desarrollo del proyecto, aportando funcionalidades y características específicas que nos han permitido un desarrollo eficiente y una implementación efectiva del sistema de inventario basado en MongoDB.

2.1. ¿Por qué MongoDB?

La elección de MongoDB para este proyecto de gestión de inventario se basa en varias ventajas significativas que ofrece esta base de datos NoSQL (No sólo SQL) frente a las bases de datos relacionales tradicionales, como MySQL. A continuación, se detallan las principales razones por las cuales hemos seleccionado MongoDB :

- **Modelo de datos flexible**: MongoDB utiliza un modelo de datos basado en documentos, donde los datos se almacenan en documentos JSON (BSON en el backend). Esto permite una gran flexibilidad en el esquema, ya que los documentos dentro de una misma colección pueden tener diferentes estructuras. En contraste, MySQL requiere un esquema fijo y predefinido, lo que puede ser limitante cuando los datos tienen estructuras variables o cuando es necesario modificar el esquema con frecuencia.

- **Escalabilidad horizontal**: MongoDB está diseñado para escalar de manera horizontal, es decir, permite distribuir la base de datos en múltiples servidores (sharding) para manejar grandes volúmenes de datos y altas cargas de trabajo. MySQL también puede escalar, pero generalmente se hace de manera vertical (aumentando la capacidad del servidor), lo cual puede ser menos eficiente y más costoso en términos de hardware.

- **Alto rendimiento y disponibilidad**: MongoDB ofrece un alto rendimiento gracias a su capacidad para almacenar datos en un formato binario JSON (BSON) que es optimizado para consultas rápidas. Además, soporta la replicación automática de datos (replica sets), lo que asegura una alta disponibilidad y recuperación ante fallos, permitiendo que la base de datos permanezca operativa incluso si uno de los servidores falla. Sí, MySQL también soporta la replicación, pero su configuración y mantenimiento son más complejos y menos automatizados.

- **Consultas y agregaciones poderosas**: MongoDB proporciona un lenguaje de consultas flexible y un framework de agregación que permite realizar operaciones complejas de manera eficiente. Las consultas pueden anidar subdocumentos y arrays, y el framework de agregación permite realizar transformaciones y análisis de datos directamente en la base de datos. MySQL utiliza SQL para sus consultas, que aunque es un lenguaje poderoso, puede ser menos intuitivo para trabajar con datos jerárquicos o anidados.

- **Desarrollo ágil**: El modelo de datos flexible de MongoDB facilita el desarrollo ágil, permitiendo iterar rápidamente sobre el diseño de la base de datos sin la necesidad de realizar migraciones complicadas. Esto es especialmente útil en proyectos donde los requisitos pueden cambiar con frecuencia o donde es necesario añadir nuevas características de manera continua. En MySQL, cambiar el esquema puede requerir alteraciones complejas, profundas

reestructuraciones del proyecto y tiempo de inactividad, lo que ralentiza el desarrollo.

- **Integración con tecnologías modernas**: MongoDB se integra fácilmente con diversas tecnologías modernas y frameworks de desarrollo, como Node.js, Express, Angular, React, etc. Esto facilita la creación de aplicaciones full-stack y permite una mejor sincronización entre la base de datos y las aplicaciones cliente. MySQL también se integra con muchas tecnologías, pero puede requerir más configuraciones adicionales para lograr una integración óptima.

- **Soporte para datos no estructurados y semi-estructurados**: MongoDB es ideal para manejar datos no estructurados y semi-estructurados, como logs, datos de sensores, y otros tipos de información que no siguen un esquema fijo. MySQL, siendo una base de datos relacional, está mejor adaptada para datos estructurados, y puede ser menos eficiente cuando se trata de datos más heterogéneos.

Elegimos pues MongoDB para este proyecto basándonos en su capacidad para manejar datos de manera flexible y escalable, su alto rendimiento, y su facilidad de integración con tecnologías modernas. Estas características hacen de MongoDB una opción ideal para la gestión de inventarios en un entorno empresarial dinámico y en constante evolución. Además, ya a título personal se nos hace efectivamente más sencillo de usar al no ser tan rígido como MySQL

3. PERSPECTIVA ESTÁTICA

3.1. Diseño de la Base de Datos: E/R

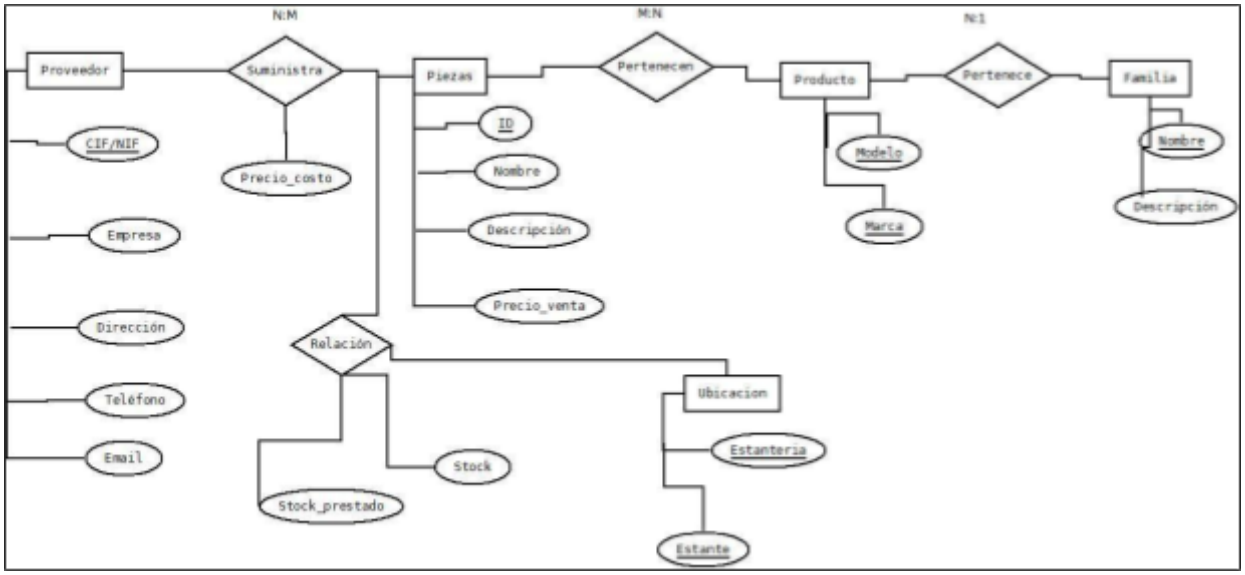


Fig. 01 Diagrama E/R principal de la BDD planteada

Mediante este primer esquema Entidad/Relación de la base de datos a utilizar para nuestra aplicación, obtenemos la información necesaria para obtener un inventario completo donde situemos tanto proveedores, piezas, productos e incluso el propio electrodoméstico, relacionando entre sí éstos elementos.

Sin embargo, a la hora de aplicar la base de datos a efectos de la aplicación a desarrollar, simplificaremos mucho la misma a fin de evitar redundancias y duplicidades innecesarias en otros departamentos de gestión de datos del taller dónde se pondrá en funcionamiento.

Por ejemplo, a efectos de control del inventario en un taller tan pequeño dónde el proveedor siempre es el mismo, no nos es necesario añadir en nuestra aplicación la opción de relacionar proveedores

3.2. Paso a patrón relacional

Dada la naturaleza sencilla de la aplicación usamos en éste caso únicamente la **incorporación (embedding)** de todos los datos (Referencia, Nombre, Cantidad, Precio, Ubicación)a la colección de piezas del inventario. Como muestra el siguiente ejemplo:

```
use inventario

db.piezas.insertOne({
    "referencia": "15101000",
    "nombre": "Motor Beta900 IX",
    "cantidad": 5,
    "precio": 55.15,
    "ubicacion": "C3"
})
```

3.3. CRUD

El acrónimo **CRUD** se refiere a las cuatro operaciones básicas que se pueden realizar en una base de datos: **Create (Crear), Read (Leer), Update (Actualizar) y Delete (Eliminar)**. Aunque MongoDB y MySQL permiten estas operaciones, su implementación y manejo difieren debido a la naturaleza de cada sistema de base de datos.

3.3.1. DDL = CREATE

Como ya sabemos, DDL (Data Definition Language) es un subconjunto de SQL que se utiliza para definir y gestionar la estructura de la base de datos. Incluyendo comandos como:

- CREATE: Para crear nuevas tablas, bases de datos, índices, vistas, etc.
- ALTER: Para modificar la estructura de objetos existentes en la base de datos.
- DROP: Para eliminar objetos de la base de datos.
- TRUNCATE: Para eliminar todos los registros de una tabla, pero manteniendo su estructura.
- RENAME: Para cambiar el nombre de un objeto de la base de datos.

En este caso, el comando CREATE es el que estaríamos comparando con MongoDB.

A continuación, se expone un ejemplo de Crear (Create) en nuestro programa:

En python:

```
pieza = {
    "referencia": "15101000",
    "nombre": "Motor Beta952",
    "cantidad": 1,
    "precio": 25.50,
    "ubicacion": "A1"
}
coleccion.insert_one(pieza)
```

En MongoDB:

```
db.piezas.insertOne({
    "referencia": "15101000",
    "nombre": "Motor Beta952",
    "cantidad": 1,
    "precio": 25.50,
    "ubicacion": "A1"
})
```

3.3.2. DQL = READ

DQL (Data Query Language) es una categoría de SQL que se enfoca en la consulta de datos en una base de datos. El comando principal de DQL es `SELECT`, que se utiliza para recuperar datos de una o más tablas.

Sin embargo, en MongoDB las operaciones READ son las acciones que recuperan documentos de las colecciones. Las operaciones principales para leer datos en MongoDB son '`find`' y '`findOne`'.

Por lo que, la manera de poder consultar en nuestra base de datos de inventario sería la siguiente:

Función de "Buscar Pieza" en Python:

```
# Búsqueda de un documento por referencia
referencia = "12345"
pieza = coleccion.find_one({"referencia": referencia})
print(pieza)
```

En MongoDB:

```
db.piezas.findOne({"referencia": "12345"})
```

3.3.3. DML = UPDATE

DML (Data Manipulation Language) es un subconjunto de SQL que se utiliza para la manipulación de datos en una base de datos relacional dónde las operaciones principales incluyen:

- INSERT: Agrega nuevas filas a una tabla.
- SELECT: Recupera datos de una o más tablas (aunque SELECT es más comúnmente clasificado bajo DQL, también puede considerarse parte de DML en algunos contextos).
- UPDATE: Modifica datos existentes en una tabla.
- DELETE: Elimina datos de una tabla.

En este caso, el comando UPDATE es el que se puede comparar con MongoDB ya que la operación UPDATE se utiliza para modificar documentos existentes dentro de una colección. MongoDB proporciona varios métodos de actualización, como '`updateOne`', '`updateMany`', y '`replaceOne`'.

En el proyecto que nos ocupa, aplicamos estos métodos de la siguiente manera:

En Python:

```
# Actualización de un documento por referencia
referencia = "12345"
nuevo_valor = {
    "$set": {
    "nombre": "Botonera L3",
    "cantidad": 15,
    "precio": 30.75,
    "ubicacion": "B2"
    }
}
coleccion.update_one({"referencia": referencia}, nuevo_valor)
```

En MongoDB:

```
db.piezas.updateOne(
    {"referencia": "12345"},
    {
    $set: {
           "nombre": "Botonera L2",
           "cantidad": 15,
           "precio": 30.75,
           "ubicacion": "B2"
    }})
```

3.3.4. DCL = ...

Así como DCL (Data Control Language) es la categoría de comandos en SQL que se utiliza para gestionar los permisos y la seguridad en una base de datos, MongoDB no dispone de un sistema de control de acceso integrado equivalente.

Sin embargo, se centra en proporcionar flexibilidad y rendimiento, delegando el control de acceso a nivel de aplicación o mediante otras herramientas externas.

Así pues, permite configurar autenticación para verificar las credenciales de usuario y autorización para controlar el acceso a la base de datos y para controlar los permisos de acceso a los recursos.

Además, también permite configurar reglas de firewall y otras medidas de seguridad para controlar el acceso a la base de datos desde las aplicaciones y otros recursos.

Por lo tanto, en el proyecto que nos ocupa, aunque no vamos a disponer de momento de roles ni usuarios, sí que lo podríamos hacer fácilmente en un futuro. Unos posibles ejemplos de controlar la creación de usuarios y roles de esta manera serían las siguientes:

1° Creamos un usuario (JoanProfe) con un rol (lectura):

```
use 'colladoinventario'

db.createUser({
  user: "JoanProfe",
  pwd: "PasswordJoanProfe",
  roles: [
    { role: "read", db: "colladoinventario" }
  ]
})
```

2° También podemos crear roles personalizados para ciertas partes de la base de datos, como el rol de oficinista:

```
use himbentario

db.createRole({
  role: "oficinista",
  privileges: [
    {
    resource: { db: "himbentario", collection: "Pieza" },
    actions: [ "find", "insert", "update", "remove" ]
    }
  ],
  roles: []
})
```

3° Y añadimos ahora el trabajador de la oficina al rol de oficinista:

```
use himbentario

db.createUser({
  user: "Anacleto",
pwd: "244466666",
  roles: [
    { role: "oficinista", db: "colladoinventario" } ] })
```

3.3.4. DELETE

De cara a eliminar datos de una tabla, en SQL vemos que el subconjunto DML contiene la instrucción DELETE.

En MongoDB, la operación para eliminar documentos se realiza principalmente a través de los métodos '`deleteOne`' y '`deleteMany`', dependiendo de si se desea eliminar un solo documento que coincida con ciertos criterios o varios documentos al mismo tiempo.

En nuestro proyecto, podremos entonces utilizar esta opción de la siguiente manera:

En Python:

```
# Eliminación de un documento por referencia
referencia = "12345"
coleccion.delete_one({"referencia": referencia})
```

En MongoDB:

```
db.piezas.deleteOne({"referencia": "12345"})
```

4. PERSPECTIVA DINÁMICA

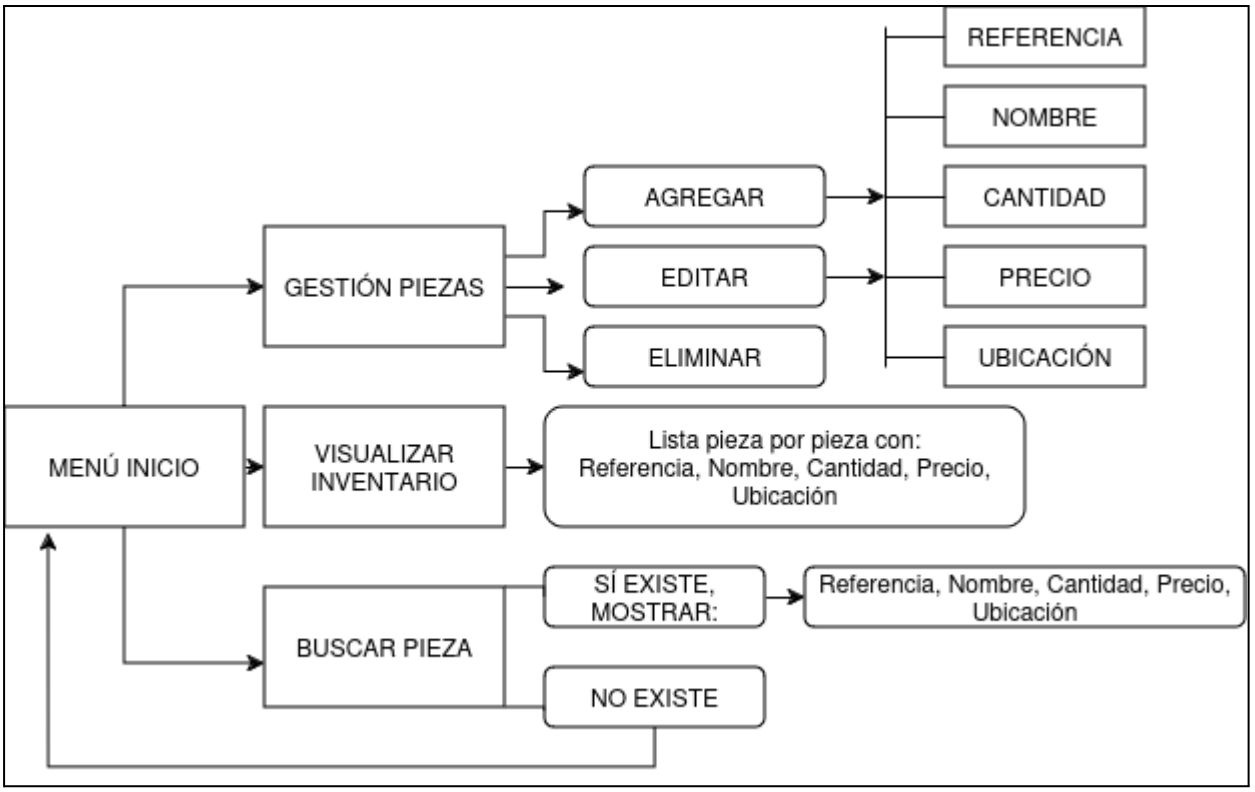


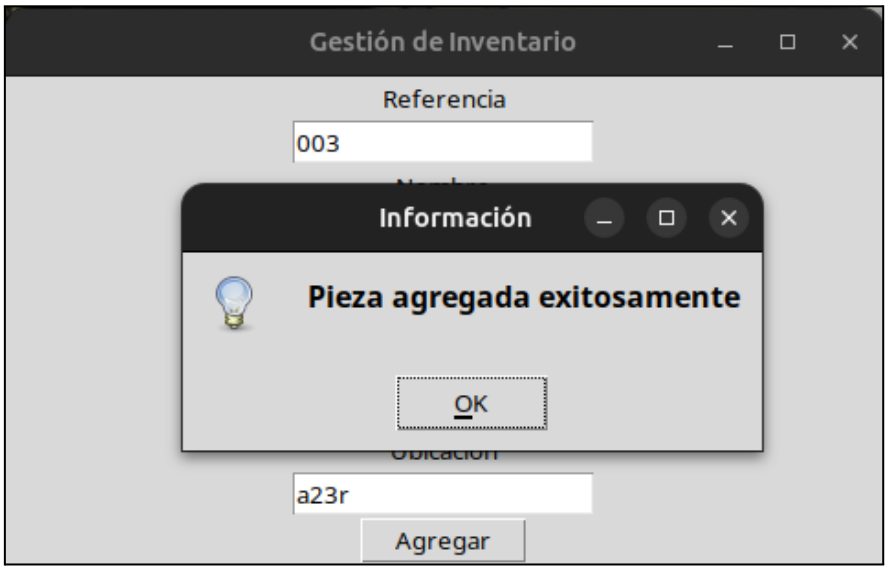
Fig. 02 Esquema casos de uso de la aplicación funcional

4.1. Casos de uso

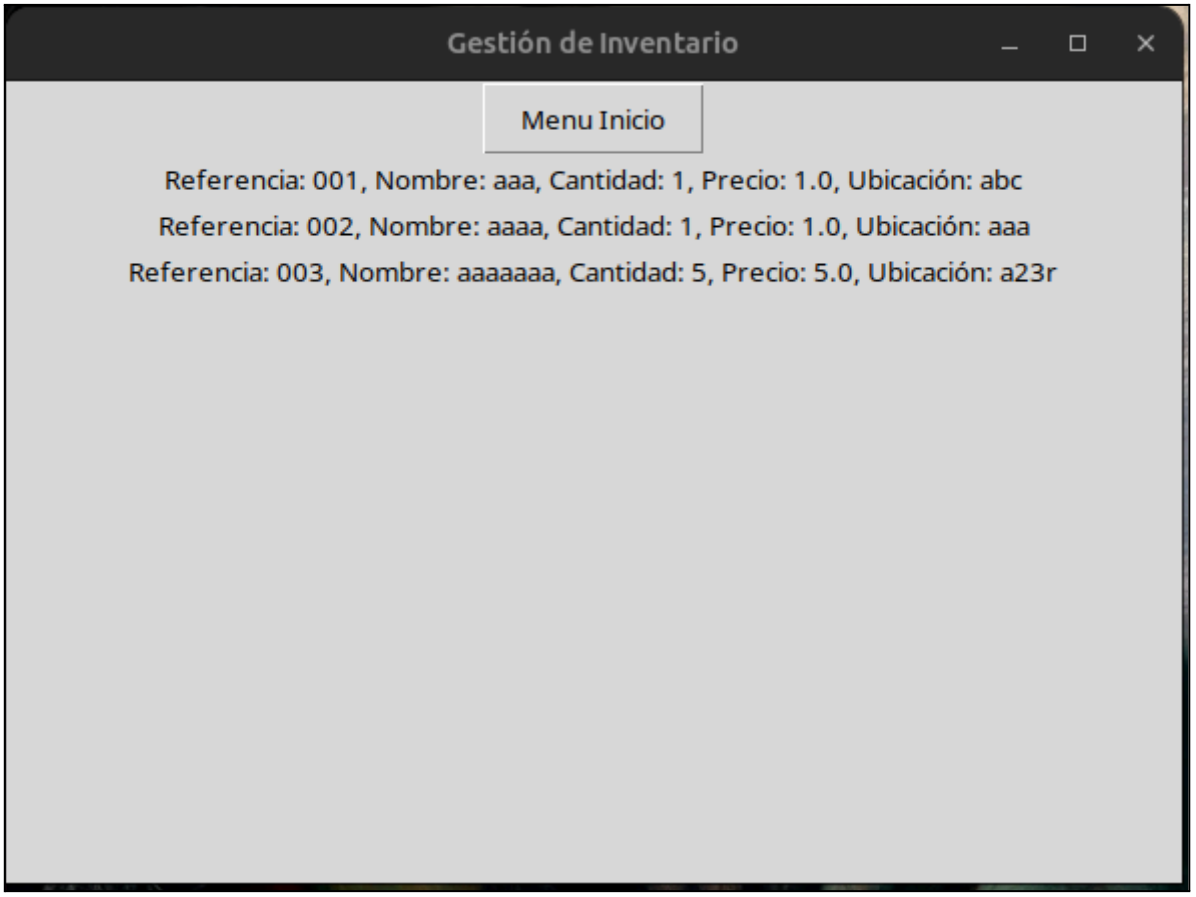
En este apartado vamos a proceder a presentar posibles casos de uso. Desde luego no serían los únicos, pero son los que más se darían.

4.1.1. Agregar pieza


Gestión de Inventario
Agregar
Editar
Eliminar
Menu Inicio


Gestión de Inventario
Referencia
003
Información
Pieza agregada exitosamente
OK
a23r
Agregar

4.1.2. Visualizar Inventario



5. CONCLUSIONES

Después de evaluar la implementación de este sistema de inventario para un taller de piezas de electrodomésticos utilizando tanto MySQL como MongoDB, se ha concluido que MongoDB podría ser la opción más adecuada. Esta decisión se basa en los puntos siguientes:

- **Flexibilidad y Evolución del Esquema:** MongoDB, al ser una base de datos NoSQL orientada a documentos, ofrece una flexibilidad excepcional en la gestión de datos. A medida que el taller crece y las necesidades de la aplicación evolucionan, es probable que la estructura de los datos cambie. MongoDB permite modificaciones en la estructura de los documentos sin necesidad de migraciones complejas, lo que facilita la incorporación de nuevas características y ajustes rápidos en el esquema de datos.

- **Escalabilidad y Rendimiento:** MongoDB está diseñado para escalar horizontalmente, lo que es ideal para un taller que anticipa un crecimiento significativo. La capacidad de distribuir datos a través de múltiples servidores permite manejar incrementos en la carga de trabajo y el volumen de datos sin comprometer el rendimiento. Esta característica es esencial para un sistema de inventario que deberá gestionar un número creciente de transacciones y datos a medida que el negocio se expande.

- **Gestión de Usuarios y Permisos:** A medida que la aplicación de inventario del taller incorpora funcionalidades adicionales como la gestión de usuarios y permisos, MongoDB ofrece una ventaja significativa. Su modelo de documentos permite almacenar datos de usuarios y permisos de manera flexible y eficiente.

- **Desarrollo Rápido y Ágil:** MongoDB facilita un desarrollo ágil gracias a su modelo de datos flexible y su integración con tecnologías modernas. Podremos iterar rápidamente, añadir nuevas características y adaptar la aplicación a nuevas necesidades sin enfrentar las restricciones de un esquema rígido. Esta agilidad es crucial para un negocio en crecimiento que necesita responder rápidamente a cambios en el mercado y en sus operaciones internas.

- **Integración y Ecosistema:** MongoDB se integra bien con una amplia gama de herramientas y servicios modernos, facilitando la incorporación de análisis de datos, monitoreo en tiempo real y otras capacidades avanzadas. Además, su compatibilidad con múltiples lenguajes de programación y frameworks permite desarrollar aplicaciones robustas y escalables, soportando tanto las necesidades actuales como futuras del taller.

En conclusión, aunque MySQL ofrece una estructura y consistencia robustas, las ventajas de flexibilidad, escalabilidad y adaptabilidad de MongoDB lo hacen más adecuado para un taller de piezas de electrodomésticos que está en expansión.

MongoDB proporciona las herramientas necesarias para gestionar el crecimiento del negocio y la complejidad de la aplicación de manera eficiente, asegurando que el sistema de inventario pueda evolucionar junto con las necesidades del taller.

A pesar de su simplicidad actual, esta aplicación proporciona una organización adicional y representa una base sólida que puede ser mejorada en el futuro.

En cuanto a preferencias personales, espero explorar más a fondo bases de datos NoSQL como MongoDB en el futuro. Inicialmente, su uso parece más intuitivo y visualmente más accesible. El uso de herramientas complementarias como Mongoose facilita aún más la integración y reduce la posibilidad de errores comunes, comparado con la rigidez y estructura más estricta de MySQL.

6. ENLACE A REPOSITORIO

A continuación se enlaza el repositorio en GitHub del Microproyecto completo:

https://github.com/CCC-Programoeba/BDD_ProyectoInventarioMongo