**Imagen que contiene Texto

Descripción generada automáticamente**

**Entrega 3 – Proyecto**

**“Sistema de Gestión de Inventario y Ventas”**

**Escuela de Ingeniería, Ciencia y Tecnología**

**Bogotá D.C., Colombia**

**Ingeniería de datos**

**Tatiana Cabrera**

**Prince Bermúdez**

**Laura Ippolito**

**Luna Márquez**

**Contenido**

[**1.** **Introducción** 3](#_Toc183378947)

[**2.** **Definición del problema** 3](#_Toc183378948)

[**3.** **Objetivos** 4](#_Toc183378949)

[**4.** **Alternativas de Solución** 4](#_Toc183378950)

[**5.** **Alcance** 5](#_Toc183378951)

[**6.** **Metodología** 5](#_Toc183378952)

[**7.** **Formato de requisitos funcionales y no funcionales** 6](#_Toc183378953)

[**8.** **Historias de usuario** 8](#_Toc183378954)

[**9.** **Diagrama de clases** 9](#_Toc183378955)

[**10.** **MER** 10](#_Toc183378956)

[**11.** **MR** 11](#_Toc183378957)

[**12.** **Diccionario de datos** 12](#_Toc183378958)

[**13.** **Base de Datos NoSQL** 13](#_Toc183378959)

[**14.** **Diagrama de procesos según la arquitectura utilizada** 14](#_Toc183378960)

[**15.** **Especificaciones de Implementación** 15](#_Toc183378961)

[**16.** **Biografías** 15](#_Toc183378962)

[**17.** **Stakeholders** 16](#_Toc183378963)

[**18.** **Conclusiones** 16](#_Toc183378964)

[**19.** **Referencias** 17](#_Toc183378965)

# **Introducción**

El proyecto “Sistema de Gestión de Inventario y Ventas” nace de la necesidad de optimizar y automatizar los procesos de gestión de inventarios y ventas en un negocio de tamaño mediano. Este sistema busca organizar la información relacionada con productos, proveedores, usuarios y ventas, permitiendo un control más preciso del stock para así generar alertas automáticas para reabastecerlos. Además, se van a implementar reportes de ventas para poder apoyar la toma de decisiones estratégicas basadas en datos.

El sistema propuesto es un ejemplo del uso de la programación orientada a objetos, esto por que las entidades clave como producto, usuario, proveedores y ventas se pueden modelar como clases con sus respectivos métodos y atributos. Cada clase está diseñada de manera modular, esto permite una mayor flexibilidad y escalabilidad a la hora de modificar el sistema. La relación entre clases refleja el principio de composición y herencia, permitiendo una mejora en la eficiencia y reutilización del código.

Este proyecto se desarrollará usando metodologías ágiles, específicamente el marco de trabajo Scrum, un marco de trabajo dentro de las metodologías ágiles, y esto permite una organización y ejecución del proyecto en varios sprints. Para seguir con el proceso del proyecto se usarán herramientas como Trello para gestionar y organizar las tareas, y GitHub para el control de los adelantos. Esta estructura ágil puede facilitar la integración de mejoras y la recepción de retroalimentación, y esto puede ayudar a ajustar la solución dependiendo de las necesidades específicas del negocio. El equipo incluye el Product Owner (PO), quien interactúa con el cliente para identificar y priorizar necesidades.

# **Definición del problema**

El problema principal que enfrenta el negocio es la ineficiencia en la gestión manual del inventario y las ventas.

Esto genera errores frecuentes en la actualización de stock, provoca que se agoten ciertos productos sin previo aviso y dificulta la toma de decisiones estratégicas debido a la falta de información actualizada.

* *Requerimientos Funcionales:*

1. Centralizar la información de inventario y ventas en una base de datos digital.
2. Automatizar las alertas de reabastecimiento de productos.
3. Generar reportes automáticos sobre las ventas y el inventario.
4. Permitir la gestión de usuarios con diferentes niveles de acceso.

* *Requerimientos No Funcionales:*

1. El sistema debe ser fácil de usar y accesible sin requerir formación técnica avanzada.
2. Debe ser escalable, permitiendo agregar más productos sin afectar el rendimiento.
3. Los datos deben estar protegidos mediante medidas de seguridad como la encriptación.

# **Objetivos**

*Objetivo General:*

Diseñar y desarrollar un sistema que automatice la gestión de inventarios y ventas, optimizando el control de stock y mejorando la eficiencia operativa del negocio.

*Objetivos Específicos:*

1. Realizar el levantamiento de información sobre las necesidades del negocio, incluyendo productos, proveedores y ventas.
2. Diseñar el modelo de base de datos relacional que centralice la información del inventario, ventas y proveedores.
3. Implementar un sistema de alertas automáticas para reabastecimiento cuando los niveles de stock bajen de cierto umbral.
4. Desarrollar un sistema de generación de reportes automáticos que permita a los gestores del negocio tomar decisiones informadas.
5. Facilitar la gestión de usuarios con diferentes niveles de acceso, garantizando la seguridad de los datos.

# **Alternativas de Solución**

Existen varias soluciones disponibles en el mercado que tienen problemas similares. Por ejemplo, software de gestión de inventario como TradeGecko y Zoho Inventory brindan características como la automatización de inventarios y la generación de reportes. Sin embargo, pueden llegar a ser costosas para pequeñas y medianas empresas.

*Solución 1:* Un sistema de inventario propio, desarrollado a medida, que incluye todas las funcionalidades requeridas por el negocio. Este sistema se desarrolla utilizando tecnologías de código abierto como MySQL y MongoDB, lo que reduce costos.

*Solución 2:* Implementación de un sistema modular que permita integrar herramientas de terceros como Stripe o QuickBooks para la gestión de pagos y contabilidad, pero manteniendo el control del inventario en una base de datos propia.

# **Alcance**

El sistema final incluirá los siguientes módulos:

1. Gestión de productos y proveedores.
2. Gestión de inventario, con alertas automáticas de reabastecimiento.
3. Registro y análisis de ventas.
4. Gestión de usuarios, con distintos niveles de acceso.

*Público objetivo:* Negocios medianos que necesiten automatizar su gestión de inventarios y ventas.

*Plataforma tecnológica:* El sistema será accesible desde navegadores web y dispositivos móviles.

# **Metodología**

Para el desarrollo del proyecto, se adoptó la metodología ágil Scrum, un marco de trabajo dentro de las metodologías ágiles, que se basa en ciclos iterativos y permite la entrega continua de valor a través de sprints. Esta metodología fue seleccionada debido a su enfoque flexible y su capacidad para adaptarse a cambios en los requerimientos del cliente durante el desarrollo del proyecto. El equipo, compuesto por un Scrum Master, un desarrollador backend y un desarrollador frontend, trabajó en coordinación para cumplir con los objetivos planteados en cada sprint.  
El equipo incluye el Product Owner (PO), quien interactúa con el cliente para identificar y priorizar necesidades.

*Roles dentro del equipo:*

* Scrum Master (Luna Márquez): Responsable de organizar los sprints, facilitar las reuniones diarias (dailys), y asegurar que el equipo mantuviera un ritmo constante de trabajo. Además, es la encargada de remover impedimentos y asegurar la alineación del proyecto con los requerimientos del cliente.
* Desarrollador Backend (Prince Bermúdez): Encargado de diseñar y desarrollar la base de datos relacional utilizando MySQL, así como la lógica del negocio que permite la gestión del inventario, ventas y usuarios. También fue responsable de implementar las alertas automáticas de reabastecimiento.
* Desarrollador Frontend (Laura Ippolito): Se encargó de la interfaz de usuario, diseñando y desarrollando una aplicación web intuitiva para gestionar las funciones de inventario, ventas y proveedores. Laura también implementó los reportes gráficos que permiten al cliente visualizar datos de ventas y tomar decisiones estratégicas.

*Proceso de desarrollo:*

El proyecto se desarrolló en tres sprints principales, cada uno con sus propios objetivos y entregables.

* Sprint 1: En este sprint, se realizó el levantamiento de información, entrevistando al cliente y los empleados del negocio para identificar las necesidades clave. El resultado de este sprint fue la definición del modelo entidad-relación (MER) y el diagrama de clases, que fueron validados por el cliente.
* Sprint 2: Durante el segundo sprint, se implementó la base de datos y se desarrollaron los módulos backend para la gestión de productos, inventarios y proveedores. También se crearon las alertas automáticas para el reabastecimiento de productos y se realizaron pruebas unitarias para garantizar la funcionalidad correcta del sistema.

*Enlaces GitHub y Trello:*

GitHub:

<https://github.com/lauraippolito28/Juguetes.git>

Trello:

<https://trello.com/invite/b/66c3804320c5bc22dde7bd7f/ATTI99856c2be43da388edbc558ccf741ba79C284648/ing-datos>

# **Formato de requisitos funcionales y no funcionales**

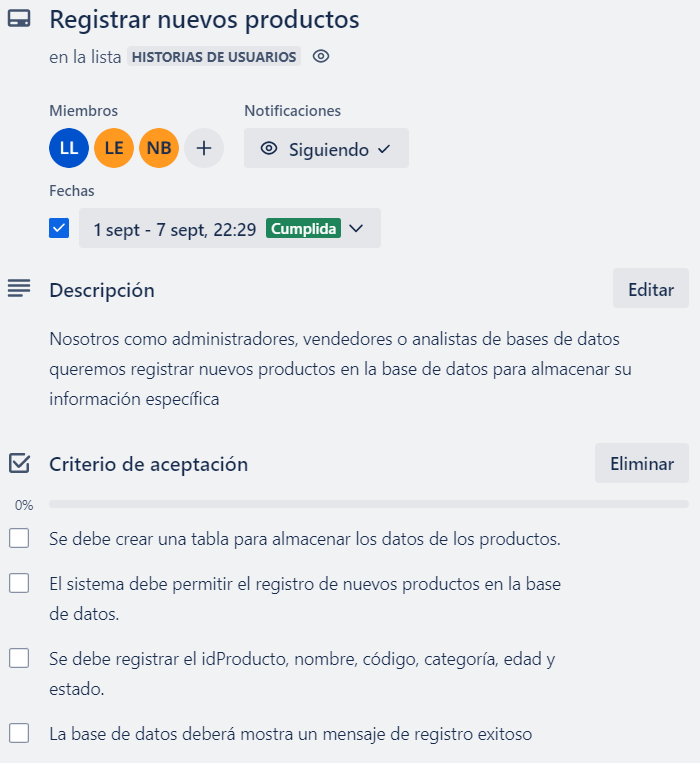
**Requerimientos Funcionales:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Código RQF** | **Nombre** | **Descripción** | **Usuario** |
| RQF001 | Centralización de información | Centralizar la información de inventario y ventas en una base de datos digital. | Administrador |
| RQF002 | Automatización de alertas | Automatizar las alertas de reabastecimiento de productos. | Administrador |
| RQF003 | Generación de reportes | Generar reportes automáticos sobre las ventas y el inventario. | Administrador |
| RQF004 | Gestión del usuario | Permitir la gestión de usuarios con diferentes niveles de acceso. | Usuario |

**Requerimientos No Funcionales:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Código RQNF** | **Nombre** | **Descripción** | **Usuario** |
| RQNF001 | Facilidad | El sistema debe ser fácil de usar y accesible sin requerir formación técnica avanzada. | Administrador |
| RQNF002 | Escalabilidad | Debe ser escalable, permitiendo agregar más productos sin afectar el rendimiento. | Administrador |
| RQNF003 | Seguridad | Los datos deben estar protegidos mediante medidas de seguridad como la encriptación. | Administrador |

# **Historias de usuario**

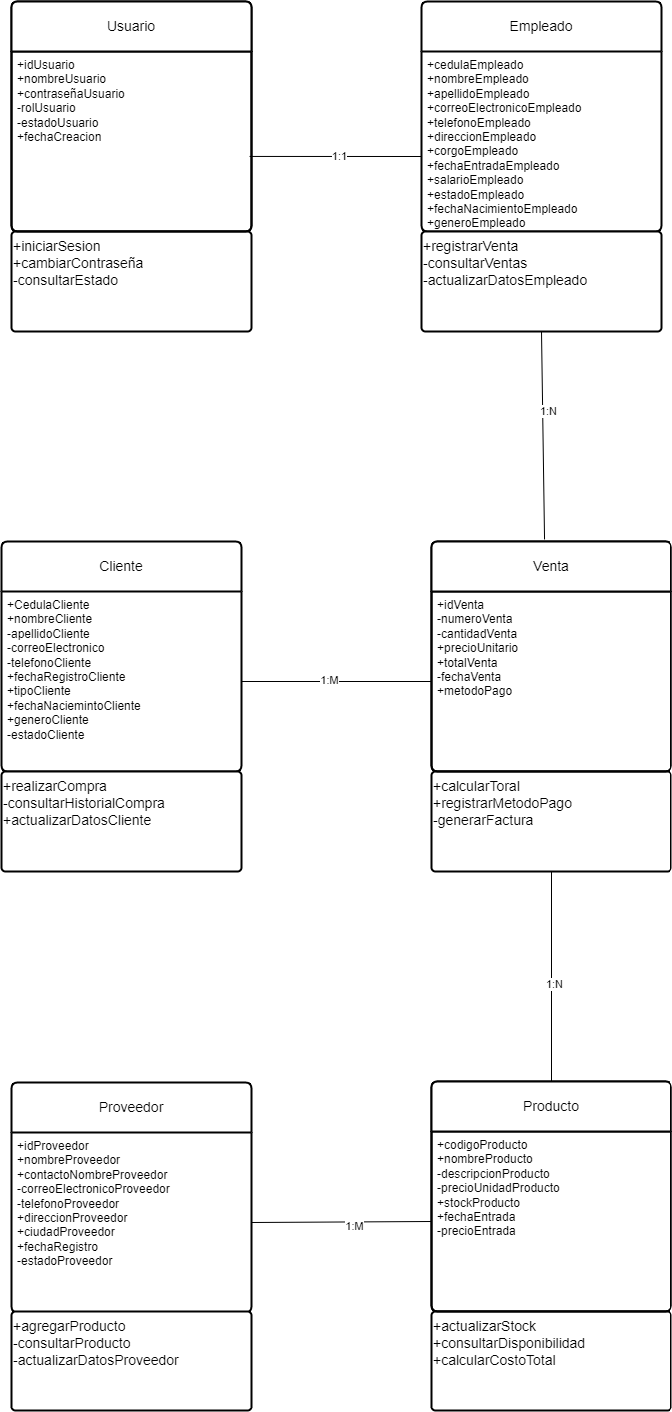
**** **Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente** Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

# **Diagrama de clases**

****

# **MER**

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

# **MR**

Interfaz de usuario gráfica, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

# **Diccionario de datos**

**Tabla

Descripción generada automáticamente**

Diccionario de datos de: **Cliente**

**Tabla

Descripción generada automáticamente**

Diccionario de datos de: **Venta**

Tabla

Descripción generada automáticamente

Diccionario de datos de: **Proveedor**

Tabla

Descripción generada automáticamente

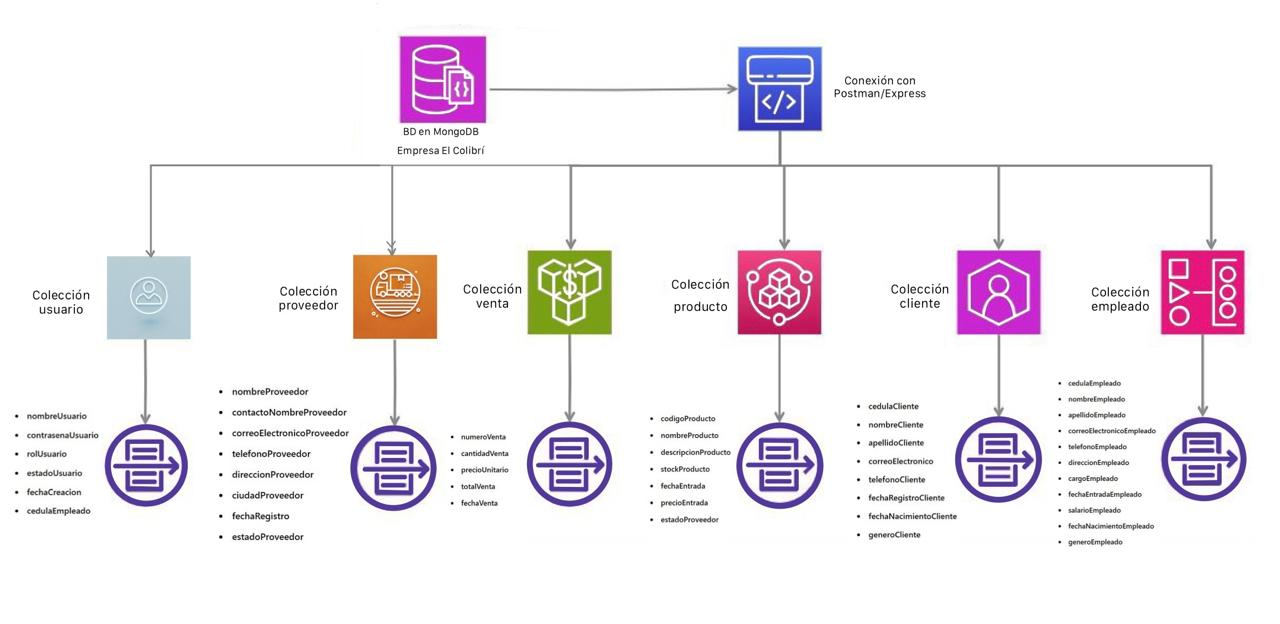
Diccionario de datos de: **Producto**

# **Base de Datos NoSQL**

La base de datos NoSQL ha sido diseñada utilizando MongoDB y Visual Studio Code. El diseño se centra en colecciones que muestran los elementos clave del sistema: empleado, cliente, proveedores, producto, usuario y venta. Cada colección fue estructurada detalladamente para así poder cumplir con los requerimientos funcionales y no funcionales establecidos desde el principio de este proyecto. MongoDB permite manejar relaciones entre colecciones, las relaciones más cruciales para el funcionamiento del proyecto son las siguientes:

* Empleado-Usuario: los empleados están vinculados a los usuarios del sistema para así poder crear un sistema de autentificación
* Venta-Cliente-Producto: Las ventas están asociadas con los clientes y los productos, esto permite generar reportes con grandes detalles del comportamiento entre cliente y el control del inventario.

Con respecto a la optimización del diseño, se utilizaron índices en campos críticos como cedulaCliente, codigoProducto y numeroVenta para poder crear un sistema más ágil y fácil de manejar con respecto a las consultas frecuentes.



***Arquitectura de la base de datos NO relacional***

# **Diagrama de procesos según la arquitectura utilizada**

Diagrama

Descripción generada automáticamente

Este diagrama de procesos muestra cómo las colecciones de la base de datos de MongoDB están bien integradas en el sistema. Con respecto al backend, se desarrolló con Express.js el cual se conecta directamente con la base de datos para poder realizar las operaciones CRUD dentro de las colecciones. Igualmente, el backend interactúa con el Postman el cual se utiliza como punto de acceso para los desarrolladores que crean este sistema. Este diseño logra facilitar una comunicación eficiente entre la lógica del negocio, la interacción del usuario y el almacenamiento de datos, esto permite que haya un flujo de información centralizado y estructurado.

# **Especificaciones de Implementación**

*1. Backend con Express y Node.js:* Se ha desarrollado una API RESTful que permite realizar operaciones CRUD sobre las colecciones de MongoDB. El backend está diseñado para manejar la comunicación entre la aplicación cliente y la base de datos. Hay tres ventajas fundamentales del diseño: escalabilidad, eficiencia y compatibilidad. [1]

*2. Mongo Compass:* Mongo Compass se usa para la visualización y gestión de la base de datos. Permite explorar las colecciones, consultar datos y verificar la estructura de los documentos. Las funcionalidades clave de este proceso son las siguientes: exploración de colecciones, consultas visuales, validación de esquemas, estadísticas de uso y gestión de índices. Con respecto a Mongo Compass hay también tres ventajas de utilizarlo: accesibilidad, optimización y depuración. [2]

*3. Mongo Shell:* Mongo Shell se emplea para ejecutar comandos de administración, crear colecciones, insertar documentos y realizar consultas avanzadas. Es una herramienta fundamental para la configuración inicial, una inserción grande datos y tareas administrativas. Las ventajas de Mongo Shell son las siguientes: flexibilidad, control detallado y eficiencia. [3]

# **Biografías**

• Prince Bermúdez es un estudiante de MACC que cada día se impone nuevos retos para sacar adelante su carrera. Su pasión son los viajes y los carros. Prince busca dejar huella en su familia siendo un buen hijo, nieto, y demás. Tiene 21 años, es de Bogotá, Colombia; su madre es de La Dorada, Caldas, y su padre es de Bogotá.

• Laura Ippolito es una estudiante de MACC con 19 años a la que le encanta viajar y descubrir nuevos lugares. Su comida favorita es la pasta, y siempre está dispuesta a probar un buen plato de espaguetis o lasaña. Además, la música es su compañera constante, ya sea para relajarse o para acompañar sus momentos de estudio y creatividad. Vive en Bogotá, Colombia, pero su familia es de Italia y le encantaría ir a vivir allá más adelante.

• Luna Márquez, estudiante de MACC, tiene 18 años y vive en Bogotá, ama cualquier actividad que la saque de la rutina como bordar o tejer, tiene un perro llamado Maní aunque es alérgica al maní, es una persona con mucha energía y ambición por sus metas.

# **Stakeholders**



# **Conclusiones**

Uno de los principales problemas que se presentó durante el desarrollo del proyecto fue la integración de los distintos módulos del sistema, especialmente en la conexión entre la base de datos relacional (MySQL) y la interfaz de usuario. Inicialmente, enfrentamos dificultades en la comunicación eficiente entre el frontend y el backend, lo que ocasionaba tiempos de respuesta lentos en ciertas consultas de inventario. Para solucionar este problema, se optimizó la estructura de las consultas SQL y se implementaron índices para mejorar el tiempo de respuesta. Además, se implementaron mecanismos de caché en ciertas operaciones de consulta para minimizar el impacto en el rendimiento.

Otro desafío importante fue la gestión de usuarios con diferentes niveles de acceso. Al implementar el sistema de autenticación, surgieron problemas relacionados con la seguridad de las contraseñas y el manejo adecuado de los permisos. Para resolver este problema, utilizamos algoritmos de encriptación para proteger las contraseñas almacenadas y diseñamos una arquitectura segura para la gestión de los roles de usuario, asegurando que cada usuario tuviera acceso solo a las funciones correspondientes a su nivel.

A nivel metodológico, el uso de Scrum, un marco de trabajo dentro de las metodologías ágiles, como metodología ágil nos permitió organizarnos mejor y mantener un flujo de trabajo constante. Sin embargo, la gestión de los tiempos en los sprints fue un reto, especialmente en el segundo sprint, donde varias tareas se retrasaron debido a problemas de integración entre los módulos. Para solventar esta situación, en los siguientes sprints se trabajó en mejorar la planificación y priorización de las tareas, logrando una entrega más eficiente en el sprint final.  
El equipo incluye el Product Owner (PO), quien interactúa con el cliente para identificar y priorizar necesidades.

Una de las mejoras que podrían implementarse en el sistema sería la integración con plataformas de comercio electrónico, lo que permitiría a los usuarios realizar ventas en línea directamente desde el sistema de gestión de inventario. Esto aumentaría la versatilidad del sistema y ofrecería más opciones al cliente para expandir su negocio.

Otra mejora potencial sería la implementación de un módulo de inteligencia artificial para realizar predicciones más precisas sobre las necesidades de reabastecimiento. Actualmente, las alertas de reabastecimiento se basan en niveles fijos de inventario, pero una IA podría analizar patrones de venta históricos y prever de manera más precisa cuándo será necesario hacer nuevos pedidos, optimizando aún más la cadena de suministro.

Finalmente, una posible mejora sería la adición de soporte multilingüe, para que el sistema pueda ser utilizado por empleados que hablen diferentes idiomas, ampliando el alcance del sistema en mercados internacionales.

Con respecto a la base de datos no relacional, donde se utilizó MongoDB, se prometió modelar las colecciones de manera eficiente, representando las entidades clave tales como empleados, clientes, productos, ventas y proveedores. La estructura fue fundamental para llegar a una gran escalabilidad del proyecto. Se diseñó bajo un enfoque modular y escalable al emplear una API RESTful que se desarrolló en Node.js con Express para así poder facilitar la comunicación entre el frontend y el backend.

También se utilizó MongoDB: MongoShell para comandos de administración y consultas más avanzadas, Mongo Compass para exploración de datos y visualización, y Visual Studio Core para desarrollar el backend. Se implementaron estrategias como la optimización de consultas y el uso de índices.

# **Referencias**

[1] Sjlouji. “Create MongoDB database models in Node.js”. Medium. Accedido el 25 de noviembre de 2024. [En línea]. Disponible: <https://javascript.plainenglish.io/node-js-models-and-database-3836f0c7f2da>

[2] “Visually Explore Your Data - Import your data”. MongoDB: The Developer Data Platform | MongoDB. Accedido el 25 de noviembre de 2024. [En línea]. Disponible: <https://www.mongodb.com/docs/compass/current/>

[3] “Run Commands - MongoDB Shell”. MongoDB: The Developer Data Platform | MongoDB. Accedido el 25 de noviembre de 2024. [En línea]. Disponible: <https://www.mongodb.com/docs/mongodb-shell/run-commands/>