

Desarrollo de una Base de Datos para un sistema de Gestión Logística de Contenedores en Almacenaje Extraportuario Aexsa

Ian José Ramírez Nina

ian.ramirez@usm.cl

Catherine Gómez B.
Profesor Guía

Resumen: Este documento plantea la implementación de una base de datos en un sistema de gestión logística de contenedores de los almacenes extraportuarios, la eficiencia de la ubicación y los movimientos de los contenedores es esencial para optimizar las operaciones logísticas. Debido al volumen de operaciones, Aexsa, una empresa dedicada al almacenaje extraportuario en San Antonio enfrenta el desafío de reducir los movimientos que realizan los operadores grúa. Esto viene relacionado con el control y ubicación de contenedores. La Base de Datos desarrollada se integra con información existente y orientada a mejorar la gestión logística de Aexsa. Se realizaron pruebas simulando la operación en el patio de contenedores. Esta solución puede ser de interés para otras empresas que gestionan contenedores en operaciones logísticas.

Palabras Clave: Contenedores, almacén extraportuario, gestión logística, base de datos.

1 Introducción

1.1 Contexto y definición del problema

El presente trabajo aborda la optimización de la gestión logística de contenedores en almacenes extraportuario, tomando como caso de estudio a Aexsa, una empresa ubicada en San Antonio y dedicada principalmente al almacenaje extraportuario. En la actualidad, Aexsa enfrenta varios desafíos logísticos relacionados con el control y la ubicación de contenedores debido al alto volumen de operaciones que se manejan. La falta de un sistema eficiente de gestión de datos provoca retrasos y el uso innecesario de las grúas, esto significa la pérdida de tiempo en la búsqueda de contenedores extraviados y al aumento de movimientos de los operadores grúas, lo que impacta negativamente en la productividad.

1.2 Propuesta de solución

Como solución a estos problemas se creó la propuesta de software 'Optainer', la cual busca abordar estos desafíos, mediante el desarrollo de una aplicación web con una visualización interactiva en 3D del patio de contenedores, esto permitirá conocer la ubicación en tiempo real de los contenedores existentes. Inicialmente se espera que funcione con verificación manual por parte de los operadores de grúa, esto sirve como base para futuras mejoras o proyectos, que puedan implementar tecnologías IoT en el futuro.

1.3 Objetivo General y Específicos

Con lo dicho, este trabajo se enfocará en el desarrollo de una base de datos para el sistema de 'Optainer', que permita mejorar el control sobre la información de los contenedores en Aexsa, teniendo en cuenta su ubicación en tiempo real, con el fin de mejorar los movimientos de los operadores de grúa.

Se establecen los siguientes objetivos específicos:

- Diseñar el modelo de datos basado en las necesidades logísticas de la empresa.
- Implementar la base de datos en un entorno adecuado, utilizando herramientas modernas de gestión de datos.
- Consultar y visualizar la información y ubicación de los contenedores.
- Validar el rendimiento y la eficacia de la base de datos mediante pruebas.

Con la implementación del sistema se logrará reducir los tiempos de búsqueda, minimizar los errores operativos y facilitar la toma de decisiones estratégicas, junto a una integración eficiente con las herramientas tecnológicas utilizadas para el sistema.

Es fundamental establecer los requerimientos funcionales que guiarán el desarrollo. Estos requerimientos permiten comprender las funciones esenciales que debe cumplir el sistema, así como las necesidades específicas relacionadas con la gestión logística de contenedores en un entorno extraportuario.

A continuación, se presenta el listado de requerimientos funcionales definidos para el desarrollo de la base de datos, considerando su integración con la visualización 3D del patio de contenedores:

Tabla 1. Listado de requerimientos funcionales

Fuente: Elaboración propia

Código de requerimiento	Nombre	Descripción
RF-1	Diseñar modelo de datos relacional	Desarrollar un modelo relacional inicial para comprender las entidades y relaciones del sistema logístico de contenedores.
RF-2	Diseñar modelo de datos NoSQL	Transformar el modelo relacional en un modelo NoSQL optimizado para la gestión y visualización de contenedores en tiempo real.
RF-3	Crear base de datos NoSQL	Construir la base de datos NoSQL para gestionar y almacenar la información de los contenedores.
RF-4	Consultas sobre los contenedores	Desarrollar consultas para obtener información de los contenedores, como posición actual y cliente asociado.
RF-5	Integración con visualización 3D.	Implementar la integración de la base de datos NoSQL con Three.js para la representación del patio de contenedores.
RF-6	Actualización en tiempo real	Garantizar que las posiciones de los contenedores se actualicen dinámicamente en la visualización 3D.

2 Marco Teórico

2.1 Fundamento de Base de Datos

Una base de datos es una herramienta para la recopilación de información que se organiza de manera sistemática. Muchas bases de datos comienzan como una hoja de cálculo, pero a medida que aumenta el volumen de información, comienza a existir redundancia e inconsistencia en los datos, por lo que, al aparecer estos problemas, la información tiene que ser transferida a una base de datos que sea controlada por un software llamado sistema de gestión de bases de datos (DBMS). Este sistema admite el control de acceso, la manipulación y la administración de los datos. Cuando se hace referencia al conjunto completo de datos, el DBMS y las aplicaciones que interactúan con ellos, a menudo se le denomina simplemente como base de datos.

2.2 Modelo de Bases de Datos

Las bases de datos pueden clasificarse en varios modelos, siendo los más comunes las bases de datos relacionales y NoSQL.

- **Bases de Datos Relacionales:** Estas estructuran la información en tablas que se relacionan entre sí mediante claves primarias y foráneas, garantizando la consistencia de los datos mediante las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Permiten manejar transacciones complejas, pero pueden limitar la flexibilidad para ciertos tipos de datos. Sistemas como MySQL y PostgreSQL son ejemplos de RDBMS que facilitan la ejecución de consultas SQL para acceder y manipular datos de manera eficiente. Tradicionalmente han sido la elección ideal para aplicaciones empresariales que requieren un alto nivel de integridad de datos y transacciones complejas.

Ventajas:

- Estructura clara y definida.
- Alta consistencia e integridad de datos.
- Atomicidad de la información.

Desventajas:

- Escalabilidad limitada.
- Menos flexibles que las NoSQL.
- Complejidad en la gestión y mantenimiento.

- **Bases de Datos NoSQL:** Estas bases de datos están diseñadas para manejar grandes volúmenes de datos no estructurados, son diseñadas para ofrecer escalabilidad y flexibilidad, por medio de las propiedades BASE (Básicamente Disponible, Estado Suave, Consistencia Eventual). A diferencia de las bases de datos relacionales, NoSQL almacena los datos en formatos variados, como documentos (MongoDB), pares clave-valor (Redis) y grafos (Neo4j). Este modelo es particularmente útil en escenarios donde se busca la velocidad de escritura y escalabilidad.

Ventajas:

- Permiten guardar datos de cualquier tipo.
- Rendimiento optimizado para cargas específicas.
- Adaptabilidad a las tecnologías actuales (IoT).

Desventajas:

- Alta curva de aprendizaje.
- Falta de estandarización.
- Falta de interfaces gráficas.

En este proyecto, es fundamental optimizar el manejo de escritura y lectura de datos debido a los cambios constantes en las ubicaciones de los contenedores. Dado que los contenedores son diariamente reubicados su ubicación varía constantemente, por lo que se necesita un sistema que ofrezca rapidez y un rendimiento optimizado. Por esas razones, se ha optado por utilizar bases de datos NoSQL.

2.3 Principios de Diseño de Bases de Datos

El diseño de una base de datos es fundamental para garantizar que los datos se gestionen de manera eficiente, segura y escalable. Un buen diseño facilita el almacenamiento, la recuperación de datos, mayor rendimiento y operaciones consistentes. La falta de un diseño adecuado puede traer problemas de redundancia, tiempos de consulta largos y de escalabilidad a medida que va creciendo el volumen de datos. A continuación, se van a presentar los beneficios y principios a seguir.

Los beneficios de tener un buen diseño:

- Reducir la redundancia de datos.
- Optimizar el rendimiento de las consultas.
- Facilitar el mantenimiento y escalabilidad.

Los principios de diseño suelen variar dependiendo del modelo a utilizar por el enfoque con el que suelen manejar la información, pero existen estrategias para asegurar el proceso de diseño. En nuestro caso en particular se verán algunos principios de diseño para las bases de datos NoSQL.

- **Eficiencia en el manejo de datos dinámicos:** Dado que se deben manejar datos en tiempo real sobre la ubicación y los movimientos de los contenedores, la base de datos debe estar diseñada para procesar un alto volumen de escrituras y lecturas de manera eficiente. El diseño se debe centrar en minimizar la latencia en las consultas, lo que es fundamental para los operadores de grúas que dependen de la información precisa y rápida para tomar decisiones.
- **Flexibilidad en la estructura de datos:** A la hora de la creación de la base de datos, es importante tener en cuenta que la estructura de datos es flexible, entonces debe ser pensada que puede estar sujeta a posibles cambios en el futuro. Para que a la hora de permitir la adición de nuevos campos no tenga que requerir cambios significativos, este principio es clave de una base de datos NoSQL.
- **Optimización de consultas específicas:** Para garantizar que las consultas sean eficientes en tiempo de respuestas para los usuarios, se debe implementar el uso de índices en las columnas, esto aumenta significativamente la velocidad de recuperación de datos, sin embargo, hay que tener cuidado con el exceso de columnas a indexar, ya que puede aumentar el tiempo de inserción y actualización de los datos.

Es importante recalcar que las bases de datos NoSQL, no sigue los mismos esquemas propios de una relacional. En su lugar, se utiliza un diseño más flexible, lo que posibilita la modelación de datos sin seguir una estructura rígida.

3 Metodología

3.1 Enfoque para el Diseño y Optimización de la Base de datos

El diseño de la base de datos para el sistema de gestión logística de contenedores en Aexsa se fundamenta en los principios de flexibilidad y escalabilidad que ofrece el modelo NoSQL. El objetivo para crear una buena estructura, siguiendo los principios de diseño antes mencionados, es crear la base de datos que permita:

- **Almacenamiento eficiente de grandes volúmenes de información:** La base de datos debe manejar datos dinámicos relacionados con la ubicación y el movimiento de los contenedores en tiempo real.

Dado que el volumen de operaciones es elevado, el sistema debe optimizar la velocidad de consulta y escritura, asegurando que la información esté disponible inmediatamente para los operadores de grúas y supervisores.

- **Adaptabilidad a cambios futuros:** Debido a que las operaciones logísticas de Aexsa pueden evolucionar con el tiempo, el diseño debe ser lo suficientemente flexible para soportar modificaciones sin alterar la estructura fundamental de la base de datos. Las bases de datos NoSQL, como MongoDB, permiten una estructura menos rígida que las relacionales, facilitando el manejo de datos no estructurados y semiestructurados.
- **Optimización del rendimiento:** Se implementan técnicas de optimización, como la creación de índices en campos clave para mejorar la velocidad de recuperación de datos y consultas frecuentes relacionadas con la ubicación y los movimientos de contenedores. Además, se aplican estrategias de particionamiento y replicación para mejorar el rendimiento y garantizar la disponibilidad de datos.

3.2 Herramientas Utilizadas

Para llevar a cabo el diseño e implementación de la base de datos, se emplearán las siguientes herramientas:

- **Modelo relacional:** Aunque las bases de datos NoSQL no requieren un esquema rígido, se utilizará un modelo relacional durante la fase inicial para modelar las entidades clave (contenedores, ubicaciones, movimientos, etc.) y sus relaciones. Este paso es importante para garantizar una visión clara del flujo de datos dentro del sistema y ayudar a definir las colecciones y documentos en MongoDB.
- **MongoDB:** Es la base de datos NoSQL seleccionada para el proyecto debido a la capacidad para manejar grandes volúmenes de datos de una forma eficiente y flexible con el manejo de datos semiestructurados. Además, MongoDB es compatible con las tecnologías utilizadas en el frontend (React) y backend (Node.js), facilitando la integración en el sistema de 'Optainer'.
- **Mongoose:** Se utilizará esta herramienta como un ORM (Object-Relational Mapping) para interactuar con MongoDB. Mongoose facilita la parte de creación y validación de esquemas, proporciona una capa adicional de control sobre los datos, así como una interfaz más estructurada para realizar consultas y operaciones sobre los documentos.

Estas herramientas en conjunto permitirán un diseño eficiente de la base de datos, optimizando el manejo de los datos en un entorno de alta movilidad y frecuencia de actualización.

3.3 Plan de Trabajo

En este proyecto se trabajará bajo la metodología ágil Scrum, que permitirá realizar de manera colaborativa los requisitos que se deben completar. Esta forma de trabajar mejora la manera que organizamos el desarrollo, en Scrum se realizan ciclos de trabajos llamados sprints, los cuales ayudan a revisar y ajustar el progreso de forma continua, normalmente un ciclo de sprint dura entre dos a cuatro semanas. En cada termino de sprint se deben entregar avances del desarrollo. Para el caso particular de este proyecto se realizaron tres ciclos de sprint de tres semanas cada uno.

3.4 Hitos Relevantes

A lo largo del proyecto, algunos hitos clave que se abordarán con respecto al desarrollo de una base de datos incluyen:

- Diseño de la base de datos.
- Implementación de la base de datos en MongoDB.
- Creación del mapa virtualizado en 3D con Three.js.
- Consumo de la base de datos desde el backend.
- Visualización de los contenedores según la base de datos.

4 Desarrollo

4.1 Diseño de la Base de Datos

El modelado de datos se basó en la estructura de un patio de contenedores en Aexsa donde cada entidad principal representa un elemento clave del sistema. La entidad 'Contenedor' almacena información única para cada unidad, mientras que la entidad 'Ubicacion' detalla sus posiciones en el mapa del patio de contenedores.

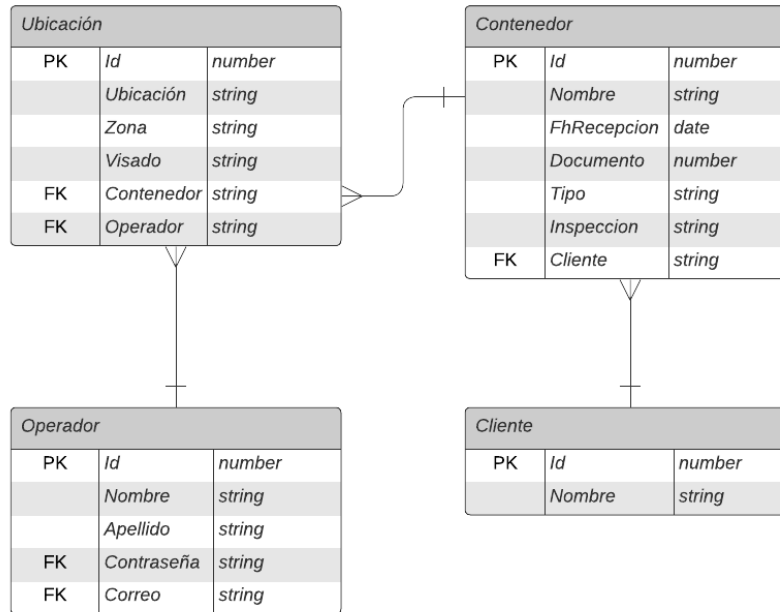


Figura 1. Modelo Relacional Optainer

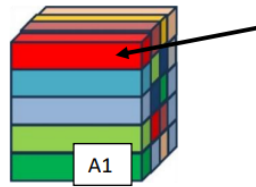
Fuente: Elaboración propia

En el sprint se define el alcance a solamente ocupar 'Ubicación', dado que proporciona la capacidad de visualizar un Contenedor existente en nuestro sistema de mapa virtualizado 3D. Además, por temas de confidencialidad no se puede mostrar la información de contenedor y clientes, pero para 'Ubicación' solo requerimos la id del Contenedor, que en este caso sería el nombre del contenedor y no su id otorgada por MongoDB. Entonces, la columna Ubicación sería de cada contenedor en el mapa 3D la cual está determinada por cuatro coordenadas principales:

- **Bloque:** Define la torre en la que se encuentra el contenedor.
- **Columna y fila:** Determinan la posición horizontal y profundidad respectivamente dentro del bloque.
- **Altura:** Especifica la cantidad de contenedores apilados en la misma posición (Z en el espacio 3D).

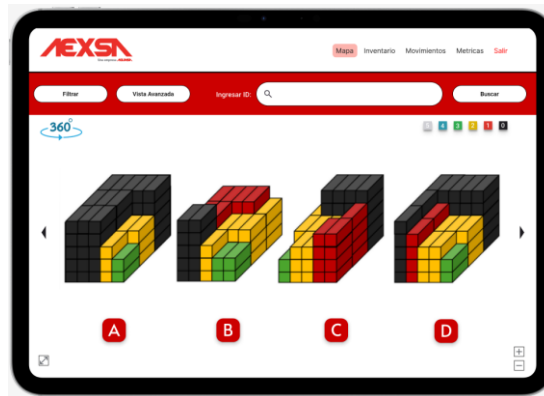
Cada una de estas coordenadas se sincroniza con el sistema de visualización utilizando un proceso de transformación de datos, donde las coordenadas de la base de datos se traducen a posiciones exactas en el espacio tridimensional.

En la figura 2 se presenta un ejemplo detallado del sistema de coordenadas utilizado en el modelo, donde se observa cómo se identifican las posiciones.

**A1-2-1-5****A:** nombre grupal del bloque**1:** sub-nombre individual dentro del grupo**2:** tamaño (1y3=20" - 2=40")**1:** profundidad, fila o pata dentro del bloque**5:** altura de abajo hacia arriba**Figura 2.** Coordenadas**Fuente:** Aexsa

Los datos se mapean a un espacio 3D generado con Three.js, donde:

- Los bloques se representan como secciones delimitadas por líneas o colores.
- Cada contenedor se renderiza como un rectángulo, cuya posición se deriva de las coordenadas bloque, fila, columna, y altura.
- El color de los contenedores varía según la altura para facilitar la identificación visual. Por ejemplo, alturas más bajas son blancas, verdes, amarillas y las más altas son rojas, negras.

**Figura 3.** Boceto mapa virtualizado 3D**Fuente:** Grupo Optainer

4.2 Implementación

Para la implementación de la base de datos, se utilizó MongoDB Atlas, un servicio en la nube que simplifica significativamente la gestión de bases de datos NoSQL al encargarse de tareas complejas como la configuración del clúster, que es el entorno donde se despliegan y gestionan las bases de datos, proporciona escalado y el monitoreo continuo. Este servicio no solo garantiza un entorno de alta disponibilidad, sino que también gestiona la seguridad al detalle, definiendo quién tiene acceso a la información y en qué condiciones. MongoDB Atlas fue la elección ideal para este proyecto debido a sus características avanzadas, que además promueven un ambiente colaborativo entre los integrantes del equipo.

Como parte de las configuraciones, cabe mencionar la existencia de una sección de MongoDB Charts, una herramienta integrada en MongoDB Atlas que propicia la creación de gráficos interactivos basados en los datos almacenados en la base de datos. Esta funcionalidad puede ser interesante para monitorear tendencias. Se puede conectar directamente las colecciones alojadas en Atlas con un entorno gráfico interactivo, eliminando la necesidad de exportar datos o usar herramientas externas. El uso de esta herramienta permite:

- **Validación de Datos:** Los gráficos verifican que las relaciones entre las entidades están correctamente definidas. Por ejemplo, la visualización de la ocupación en el patio ayuda a identificar posibles inconsistencias en las ubicaciones.
- **Toma de Decisiones:** La información generada a partir de los gráficos, puede ser de utilidad para la optimización del diseño de la base de datos, asegurando un mejor rendimiento en el manejo de grandes volúmenes de datos.
- **Presentaciones y Reportes:** Los gráficos creados en Charts pueden ser utilizados para generar reportes visuales claros y comprensibles, facilitando la presentación de resultados a los stakeholders del proyecto.

Comenzando con el proceso de configuración, lo primero es la creación de un clúster dedicado en MongoDB Atlas. Esto permitió establecer un entorno donde los datos están disponibles de manera continua. En esta sección se creó la base de datos 'optainer', en donde se pueden agregar las entidades, pero como es una base de datos NoSQL, en este caso se llaman colecciones, el nombre de cada colección ya está definido anteriormente en nuestro diagrama, solo se agregaron en plural, por la razón que una colección generalmente almacena múltiples documentos del mismo tipo. Usar un nombre en plural, como ubicaciones, indica que contiene muchas ubicaciones, en lugar de un único documento llamado ubicación, lo que resulta más intuitivo, ya que el plural indica un grupo de datos relacionado y mejora el contexto y legibilidad para el desarrollador Backend.

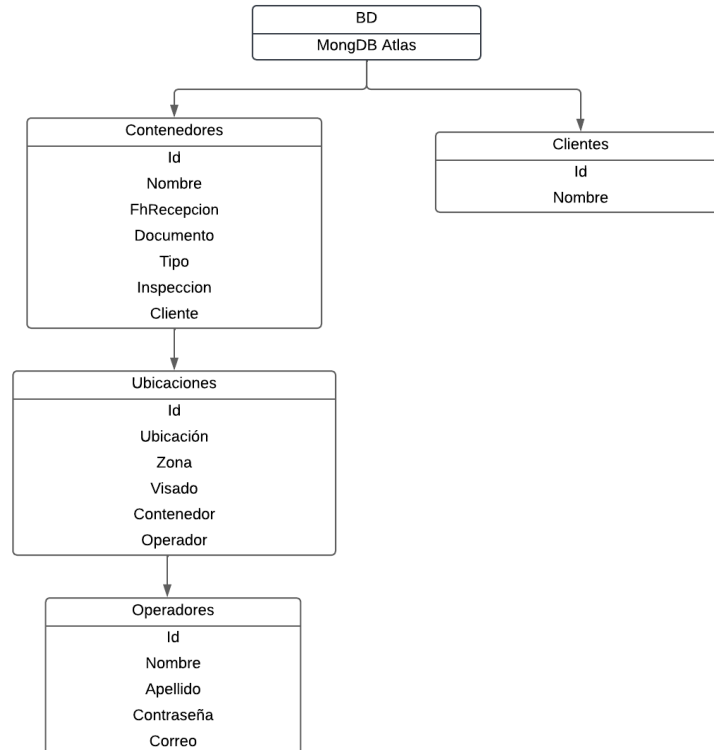


Figura 4. Modelo Jerárquico de la base de datos optainer

Fuente: Elaboración propia

A continuación, se describen los atributos de cada colección, organizados en tablas. Cada atributo se presenta junto con una breve descripción de su función dentro del sistema.

- Clientes:

Tabla 2. Atributos de la colección clientes

Fuente: Elaboración propia

Campo	Descripción
Id	Identificador único de MongoDB del cliente.
Nombre	Nombre del cliente.

- Contenedores:

Tabla 3. Atributos de la colección contenedores

Fuente: Elaboración propia

Campo	Descripción
Id	Identificador único de MongoDB del contenedor.
Nombre	Nombre del contenedor.
FhRecepcion	Fecha de recepción del contenedor.
Documento	Número de documento relacionado al contenedor.
Tipo	Código que define el tamaño y categoría del contenedor (por ejemplo, refrigerado, estándar, etc.).
Inspección	Tipo de inspección realizada al contenedor.
Cliente	Referencia al cliente asociado al contenedor.

- Ubicaciones:

Tabla 4. Atributos de la colección ubicaciones

Fuente: Elaboración propia

Campo	Descripción
Id	Identificador único de MongoDB de la ubicación.
Ubicación	Coordenada que indica la posición específica del contenedor.
Zona	Tipo de zona dentro del área de almacenamiento.
Visado	Estado de validación o autorización del contenedor.
Contenedor	Referencia al contenedor asociado a la ubicación.
Operador	Referencia al operador responsable de mover el contenedor a la ubicación.

- Operadores:

Tabla 5. Atributos de la colección operadores

Fuente: Elaboración propia

Campo	Descripción
id	Identificador único de MongoDB del operador.
Nombre	Nombre del operador.
Apellido	Apellido del operador.
Contraseña	Contraseña utilizada por el operador para acceder al sistema.
Correo	Correo electrónico del operador.

Con los datos de prueba proporcionados por Aexsa, se procedió a poblar las colecciones de la base de datos. Entre estas, se destacó la carga inicial de la colección de 'ubicaciones', donde se registraron los datos necesarios para representar las posiciones de los contenedores. Esta información es fundamental para alimentar la visualización del mapa 3D del sistema, que veremos más adelante.

```
_id: ObjectId('674614aadcc8508a4c4271ba')
Contenedor : "AAAA-324234-2"
Ubicación : "AEXSA"
Zona : "COMERCIAL"
Visado : false
```

```
_id: ObjectId('674614aadcc8508a4c4271bb')
Contenedor : "MSCU-494618-4"
Ubicación : "Y2-02-01-01"
Zona : "COMERCIAL"
Visado : false
```

Figura 5. Documentos creados en la colección 'ubicaciones'

Fuente: Elaboración propia

Siguiendo con el proceso de configuraciones de MongoDB Atlas, que proporciona herramientas específicas para trabajar de manera eficiente en equipo, se tiene:

- **Gestión de Organización:**

- Se configuró una organización en MongoDB Atlas, permitiendo centralizar la administración del proyecto y añadir usuarios con distintos niveles de permisos.
- Los integrantes del equipo fueron invitados a la organización, lo que le dio acceso al clúster según las funciones asignadas.

Project Access Manager

Manage access to this project for users, teams, and Applications.

Users			
Applications			
Teams			
Find a user			
Display Name	Email Address	Project Role	Created
Charts User	charts+67240a398fbabb01fd5cc1dd@mongodb.com	Project Charts Admin	10/31/24 - 08:15:52 PM
Ian Ramirez	ian-ramirez11@hotmail.com	Project Owner	10/31/24 - 07:52:15 PM
Matias Ojeda	matias.ojeda.saa@gmail.com	Project Data Access Admin, Project Data Access Read Write	11/05/24 - 07:07:15 PM

Figura 6. Gestión de la organización

Fuente: Elaboración propia

- **Control de Acceso Granular:**

- Se establecieron credenciales específicas para el desarrollador Backend, con roles ajustados a sus responsabilidades, tales como readWrite y clusterAdmin para administradores.
- Esto mejora el control granular del acceso a los datos, garantizando la integridad de la información y evitar modificaciones accidentales.

- **Restricciones por IP:**

- Para reforzar la seguridad, se configuraron reglas de acceso basadas en direcciones IP, debido a que se está trabajando en un ambiente de desarrollo se permite el acceso desde IPs específicas, como la dirección del backend provisional y las redes de los integrantes del equipo.

Luego de todas las implementaciones, se procedió a garantizar el correcto funcionamiento de las consultas y los procesos de manejo de datos, asegurando una interacción eficiente con la base de datos en MongoDB Atlas. Para lograrlo, se desarrollaron consultas que serán utilizadas por el servicio backend, el cual posibilita realizar las operaciones principales del sistema, tales como la búsqueda de contenedores, la actualización de contenedores y la eliminación de registros cuando sea necesario.

Primeramente, se realizó la conexión del backend con MongoDB Atlas, se utilizó la biblioteca Mongoose, que facilita la interacción con bases de datos MongoDB con Node.js al proporcionar una capa de abstracción basada en esquemas. Se configuraron las credenciales necesarias para acceder a la base de datos de forma segura, las cuales fueron otorgadas por MongoDB Atlas al crear la organización y el clúster. En el archivo de configuración principal del backend, se utiliza la variable de entorno DB_CONNECTION para asegurar que las credenciales sean gestionadas de manera segura. El código que implementa esta conexión se muestra en la Figura 7.

```
1  const mongoose = require('mongoose');
2
3  mongoose.connect(process.env.DB_CONNECTION)
4    .then(() => {
5      console.log("Conexión exitosa a MongoDB Atlas")
6    })
7    .catch(err => {console.log("Error al conectar con MongoDB Atlas:", err)});
8
```

Figura 7. Código de conexión a MongoDB

Fuente: Elaboración propia

Al iniciar el servidor, se ejecuta esta configuración y se valida la conexión con MongoDB Atlas. Si la conexión es exitosa, el backend muestra el mensaje:

```
> optainer@1.0.0 start
> node index.js

Servidor corriendo en el puerto 5000
Conexión exitosa a MongoDB Atlas
```

Figura 8. Mensaje de inicio del servidor y conexión exitosa a MongoDB Atlas

Fuente: Elaboración propia

El backend se encuentra correctamente enlazado con la base de datos de “optainer”, lo que establece la base para realizar consultas y operaciones de manejo de datos. Sin embargo, para interactuar eficazmente con la colección de ubicaciones, fue necesario crear un modelo utilizando Mongoose. Este modelo se basa en un esquema que define la estructura de los documentos almacenados en la colección.

El esquema no solo garantiza la consistencia de los datos, sino que también simplifica la implementación de operaciones CRUD (Crear, Leer, Actualizar, Eliminar) al proporcionar una interfaz clara y bien definida para trabajar con la base de datos. A continuación, se presenta el esquema del modelo Ubicaciones en la figura 9.

```
1  const mongoose = require('mongoose');
2
3  const ubicacionSchema = new mongoose.Schema({
4    Contenedor: String,
5    Ubicación: String,
6    Zona: String,
7    Visado: Boolean,
8  });
9
10 module.exports = mongoose.model('ubicaciones', ubicacionSchema);
```

Figura 9. Esquema del modelo de la colección 'ubicaciones'**Fuente:** Elaboración propia

Con el modelo definido, se desarrollaron los controladores necesarios para interactuar con la base de datos. Los controladores encapsulan la lógica para realizar operaciones de consulta de datos, como búsqueda, adición, y eliminación de registros. Algunos ejemplos incluyen:

```
1  const Ubicacion = require('../models/ubicacion');
2
3  const getContenedores = async (req, res) => {
4    try {
5      const ubicaciones = await Ubicacion.find();
6      const filteredData = ubicaciones.map((ubicacion) => ({
7        contenedor: ubicacion.Contenedor,
8        ubicacion: ubicacion.Ubicación,
9        zona: ubicacion.Zona,
10       visado: ubicacion.Visado,
11     }));
12     res.json(filteredData);
13   } catch (error) {
14     res.status(500).json({ error: 'Error al obtener los datos de MongoDB' });
15   }
16   };
17 }
```

Figura 10. Controlador para obtener datos**Fuente:** Elaboración propia

```
1  const addContenedor = async (req, res, io) => {
2    const { contenedor, ubicacion, zona = '', visado = false } = req.body;
3
4    try {
5      const nuevaUbicacion = new Ubicacion({
6        Contenedor: contenedor,
7        Ubicación: ubicacion,
8        Zona: zona,
9        Visado: visado
10     });
11
12     const savedUbicacion = await nuevaUbicacion.save();
13     console.log(savedUbicacion);
14
15     io.emit('contenedorActualizado', { tipo: 'agregar', contenedor: savedUbicacion });
16
17     res.status(201).json({ message: 'Contenedor agregado a MongoDB', data: savedUbicacion });
18   } catch (error) {
19     console.error("Error al guardar en MongoDB:", error); // Log para diagnosticar el error
20     res.status(500).json({ error: 'Error al agregar el contenedor en MongoDB' });
21   }
22   };
23 }
```

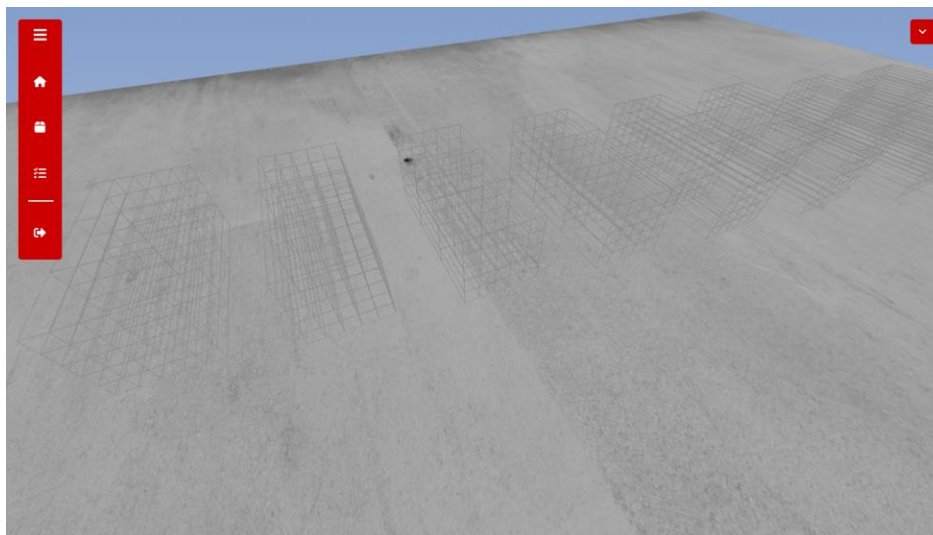
Figura 11. Controlador para agregar datos**Fuente:** Elaboración propia

```
1 const removeContenedor = async (req, res, io) => {  
2   const { id } = req.params;  
3  
4   try {  
5     const deletedUbicacion = await Ubicacion.findOneAndDelete({ Contenedor: id });  
6  
7     if (!deletedUbicacion) {  
8       return res.status(404).json({ message: 'Contenedor no encontrado' });  
9     }  
10  
11     io.emit('contenedorActualizado', { tipo: 'eliminar', id });  
12  
13     res.json({ message: 'Contenedor eliminado de MongoDB' });  
14   } catch (error) {  
15     console.error("Error al eliminar en MongoDB:", error);  
16     res.status(500).json({ error: 'Error al eliminar el contenedor en MongoDB' });  
17   }  
18 };
```

Figura 12. Controlador para eliminar datos**Fuente:** Elaboración propia

Una vez desarrollados los controladores para realizar operaciones CRUD sobre la colección de ubicaciones, los datos almacenados en la base de datos fueron utilizados para alimentar la visualización en el mapa 3D. Este mapa, diseñado en paralelo, fue integrado con el sistema para consumir las ubicaciones de los contenedores, permitiendo representar de manera gráfica y en tiempo real la disposición de estos dentro del sistema. Gracias a esta funcionalidad, se facilita la gestión, supervisión y monitoreo eficiente del patio de almacenamiento.

Al inicio, el mapa 3D mostraba únicamente una estructura base, representada por una grilla que delinea las posiciones y bloques disponibles, pero sin una representación visible de los contenedores. Este estado reflejaba la ausencia de datos cargados en la base de datos. Posteriormente, al ejecutar los controladores y consumir los datos almacenados en la colección "ubicaciones", el mapa fue actualizado de forma dinámica, permitiendo visualizar cada contenedor en su posición asignada dentro de las grillas. Esta implementación cumple con el objetivo de proporcionar una experiencia visual clara y funcional, ofreciendo una representación detallada, precisa y actualizada del estado del sistema logístico.

**Figura 13.** Mapa 3D inicial sin datos cargados**Fuente:** Elaboración propia

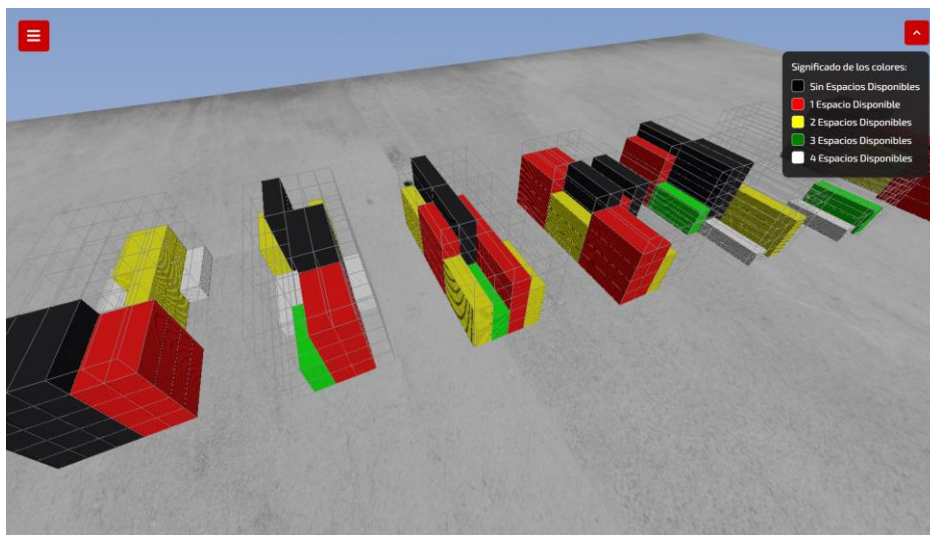


Figura 14. Mapa 3D después de cargar los datos

Fuente: Elaboración propia

Una vez cargados los datos sobre la ubicación de los contenedores en el mapa 3D, se implementó en otra vista el uso de los demás controladores, permitiendo realizar operaciones adicionales como la actualización, eliminación y consulta detallada de las ubicaciones dentro del sistema. Esto garantizó una interacción integral con los datos almacenados en la base de datos. Además, para asegurar que los cambios realizados en tiempo real se reflejen de manera inmediata en la visualización, se utilizaron sockets.

Los sockets, o puntos de conexión, son una tecnología que hace posible la comunicación bidireccional entre el cliente y el servidor, manteniendo una conexión activa que posibilita la transferencia continua de datos. En este caso, los sockets fueron clave para actualizar dinámicamente el estado del mapa 3D cada vez que se modificaba la información en la base de datos, logrando una experiencia completamente en vivo para los usuarios que utilizarán el sistema.

INVENTARIO AVANZADO

TORRE A

5	ID: HPCF-183199-9 Ubicación: A1-02-01-05	ID: HPCF-183199-2 Ubicación: A1-02-02-05	ID: EEEE-111111-7 Ubicación: A1-02-03-05	Vacio	Vacio
4	ID: ZRAF-183300-2 Ubicación: A1-02-01-04	ID: ZRAF-183300-3 Ubicación: A1-02-02-04	ID: HPCF-083002-1 Ubicación: A1-02-03-04	ID: IIII-111111-9 Ubicación: A1-02-04-04	ID: AAAA-111111-9 Ubicación: A1-02-05-04
3	ID: ZRCF-183392-2 Ubicación: A1-02-01-03	ID: LDGN-232393-7 Ubicación: A1-02-02-03	ID: HPCF-183102-8 Ubicación: A1-02-03-03	ID: HPCF-183102-9 Ubicación: A1-02-04-03	ID: HPCF-083707-1 Ubicación: A1-02-05-03
2	ID: ZRCF-183392-1 Ubicación: A1-02-01-02	ID: MSCU-528135-5 Ubicación: A1-02-02-02	ID: QBJV-161208-3 Ubicación: A1-02-03-02	ID: HPCF-183102-4 Ubicación: A1-02-04-02	ID: HPCF-183102-6 Ubicación: A1-02-05-02
1	ID: MSCU-328115-3 Ubicación: A1-02-01-01	ID: MZCP-915101-9 Ubicación: A1-02-02-01	ID: PPJV-183208-3 Ubicación: A1-02-03-01	ID: EEEE-111111-2 Ubicación: A1-02-04-01	ID: HPCF-183102-1 Ubicación: A1-02-05-01
	1	2	3	4	5

Figura 15. Vista del inventario avanzado

Fuente: Elaboración propia



INVENTARIO AVANZADO



Figura 16. Representación del sistema tras agregar un contenedor a una ubicación
Fuente: Elaboración propia



INVENTARIO AVANZADO



Figura 17. Representación del sistema tras eliminar un contenedor de una ubicación
Fuente: Elaboración propia



Figura 18. Vista de la base de datos con el contenedor manipulado en la colección de ubicaciones
Fuente: Elaboración propia

4.3 Gestión de Datos.

La gestión de grandes volúmenes de datos es una de las principales consideraciones en el desarrollo de sistemas logísticos de contenedores. Dado que los datos generados en tiempo real pueden ser masivos, es esencial contar con estrategias que aseguren el procesamiento eficiente y escalable de la información. Aunque en esta implementación inicial no se han aplicado todas las posibles soluciones, se han seguido patrones de diseño que incorporan diversas estrategias de manejo de datos a medida que el sistema escale.

El uso de una base de datos NoSQL, como MongoDB Atlas, mejora la escalabilidad y el rendimiento en la gestión de grandes volúmenes de datos no estructurados, debido a que es un servicio en la nube, a medida que se necesite más capacidad, este servicio admite el aumento de los recursos. Esta solución, además ofrece una expansión más flexible de los datos, superando las limitaciones de las bases de datos relacionales, lo que la hace ideal para el almacenamiento de ubicaciones y para futuras incorporaciones, como los movimientos de contenedores.

A continuación, se detallan algunas de las estrategias que podrían implementarse en el futuro para garantizar una mayor eficiencia en el manejo de grandes volúmenes de datos:

- **Particionamiento de Datos:**

- Con el crecimiento continuo de la cantidad de datos, se podría implementar un particionamiento de datos en base a características de los contenedores y su ubicación como la fecha o la posición del bloque que se encuentra. Este enfoque permitiría segmentar los datos en diferentes particiones para optimizar las consultas y mejorar el rendimiento general del sistema.
- Posible implementación futura: Se podría particionar los datos de la ubicación, es decir obtener la coordenada por bloque o por periodos (mensuales o trimestrales) para reducir el tiempo de consulta y la carga en el sistema.

- **Procesamiento por Lotes (Batch Processing):**

- En lugar de actualizar los datos de manera continua, se podría implementar el procesamiento de datos en lotes, especialmente para las actualizaciones que no requieren tiempo real, como los movimientos históricos de contenedores. Esto permitiría procesar grandes volúmenes de datos durante las horas de menor actividad, reduciendo el impacto en el sistema.
- Posible implementación futura: A medida que el sistema crezca, los procesos de actualización de los contenedores pueden optimizarse a través de tareas programadas que actualicen masivamente los datos a intervalos específicos.

La gestión eficiente de los datos requiere tanto seguridad como integridad, estos son aspectos fundamentales en cualquier sistema. En el contexto de este sistema, se han implementado diversas estrategias para garantizar la protección de la información y a medida que el sistema evolucione, se planifica implementar mecanismos de respaldo.

- **Protección de datos sensibles:** Se ha puesto especial atención en la seguridad y la protección de la información. El sistema no solo asegura la integridad de los datos, sino también su confidencialidad. Ya se ha mencionado el uso de roles y permisos de acceso, los cuales restringen quién puede ver o modificar los datos sensibles.
- **Respaldo:** Para evitar la pérdida de datos, es importante tener sistemas que hagan copias de seguridad de la base de datos de forma automática. Estas copias se guardan en lugares seguros, como en la nube o en servidores locales, lo que asegura que los datos estarán disponibles incluso si ocurre un problema o un fallo en el sistema. Aunque la capa gratuita de MongoDB Atlas no incluye la opción de respaldos automáticos, esta funcionalidad podrá implementarse a medida que el sistema crezca y se requieran más recursos.

4.4 Optimización y Rendimiento

La optimización de consultas es un aspecto fundamental para garantizar que el sistema sea capaz de manejar grandes volúmenes de datos de manera eficiente. Para lograrlo, se emplearon diversas técnicas, tales como:

- **Uso de consultas específicas:** En lugar de realizar consultas generales que podrían devolver grandes cantidades de datos innecesarios, se diseñaron consultas que filtran solo los datos relevantes. Esto mejora la eficiencia al reducir el volumen de información procesada.

```
1  const filtrarContenedor = () => {
2    const contenedorEncontrado = filtrarContenedorPorId(data, contenedorId);
3    if (contenedorEncontrado) {
4      setTorreActual(contenedorEncontrado.ubicacionParseada.torre);
5      setProfundidadActual(contenedorEncontrado.ubicacionParseada.z);
6      setContenedorResultado(contenedorId);
7      console.log("Contenedor encontrado:", contenedorEncontrado);
8    } else {
9      alert("Contenedor no encontrado");
10     console.warn("Contenedor no encontrado con ID:", contenedorId);
11   }
12  };
```

Figura 19. Consulta específica de un contenedor

Fuente: Elaboración propia

- **Paginación de resultados:** Cuando es necesario recuperar grandes conjuntos de datos, la paginación carga solo una parte de los resultados a la vez, lo que reduce la carga de la base de datos y mejora la experiencia del usuario al evitar tiempos de espera prolongados.

```
1  const contenedoresFiltrados = data.filter((contenedor) => {
2    if (contenedor.ubicacionParseada) {
3      return contenedor.ubicacionParseada.torre === torreActual &&
4        contenedor.ubicacionParseada.z === profundidadActual;
5    }
6    return false;
7  });
```

Figura 20. Filtrado de datos por bloque y profundidad

Fuente: Elaboración propia

Los índices y particionamiento son dos herramientas fundamentales para mejorar el rendimiento de la base de datos, especialmente cuando se manejan grandes volúmenes de información.

- **Índices:** Mongoose permite la creación de índices en los campos que son consultados con mayor frecuencia, lo que mejora considerablemente la eficiencia de las búsquedas. Al usar índices, la base de datos no necesita recorrer toda la colección para encontrar un documento, lo que reduce el tiempo de respuesta de las consultas. Para este sistema, se diseñaron índices en campos clave como ubicación y contenedor, lo que optimiza las consultas relacionadas con estos datos y asegura una búsqueda rápida y eficiente. En la Figura 21, se muestra un ejemplo de un índice compuesto único en los campos ubicación y contenedor. Este índice garantiza que no existan documentos duplicados con la misma combinación de estos dos campos, lo que asegura la integridad de los datos en la colección.



```
1  const mongoose = require('mongoose');
2
3  const ubicacionSchema = new mongoose.Schema({
4    Contenedor: String,
5    Ubicación: String,
6    Zona: String,
7    Visado: Boolean,
8  });
9
10 ubicacionSchema.index({ Ubicación: 1, Contenedor: 1 }, { unique: true });
11
12 module.exports = mongoose.model('ubicaciones', ubicacionSchema);
```

Figura 21. Esquema con índice compuesto del modelo de la colección "ubicaciones"

Fuente: Elaboración propia

- **Particionamiento (Sharding):** El particionamiento de la base de datos es debido al uso de la capa gratuita de MongoDB Atlas, este distribuiría los datos entre varios nodos del cluster, donde cada nodo es responsable de una parte de los datos, esto mejora el rendimiento de las consultas y la escalabilidad.

4.5 Resultados y Discusión

El objetivo principal de este proyecto fue desarrollar una base de datos eficiente para el sistema 'Optainer', que permita mejorar el control sobre la información de la ubicación de los contenedores de Aexsa, con un enfoque en la ubicación en tiempo real. Para evaluar el rendimiento de la base de datos, se llevaron a cabo pruebas que medían la eficiencia de las consultas, la rapidez en las actualizaciones de las ubicaciones, y la capacidad de manejar grandes volúmenes de datos. Los resultados fueron positivos en los siguientes aspectos:

- **Consultas rápidas:** Las consultas sobre la ubicación de los contenedores en tiempo real fueron rápidas, incluso cuando la base de datos fue creciendo en tamaño de documentos. Esto se debe a la implementación de índices en campos clave como la ubicación y el contenedor, lo que permitió recuperar rápidamente la información necesaria.
- **Actualizaciones eficientes:** Las operaciones de actualización, que simulan los movimientos de los contenedores y su reubicación, agregando y eliminando, fueron realizadas de manera eficiente sin generar demoras significativas en el sistema, lo cual es crucial para un sistema logístico en tiempo real.
- **Pruebas de carga:** Se realizó una simulación de carga que reflejó el comportamiento del sistema bajo condiciones de alta demanda. Los resultados demostraron que la base de datos puede manejar un aumento en el volumen de contenedores y consultas sin impactar significativamente el rendimiento.

En general, la base de datos fue capaz de manejar las necesidades operativas de la empresa Aexsa con una alta eficiencia, mejorando la visibilidad y el control sobre las ubicaciones de los contenedores y permitiendo tomar decisiones más informadas.

Durante las pruebas de rendimiento, se evaluaron las métricas proporcionadas por MongoDB Atlas, incluyendo el número de conexiones activas, la cantidad de datos transferidos y el número de operaciones realizadas por segundo.

Los resultados reflejan un bajo número de conexiones activas en condiciones normales, con picos ocasionales durante simulaciones de alta carga, lo que indica una gestión eficiente de las consultas. La transferencia de datos se mantuvo en niveles mínimos durante la mayoría de las pruebas, demostrando que las consultas fueron optimizadas para recuperar únicamente los datos necesarios. En cuanto al OpCounter, los resultados mostraron una tasa moderada de operaciones por segundo, con valores ocasionalmente variables entre los nodos del cluster. Aunque no se alcanzaron cifras extremadamente altas, la consistencia en la velocidad de las operaciones sugiere que la base de datos es capaz de manejar eficientemente consultas, actualizaciones y otras transacciones sin comprometer el rendimiento general del sistema.

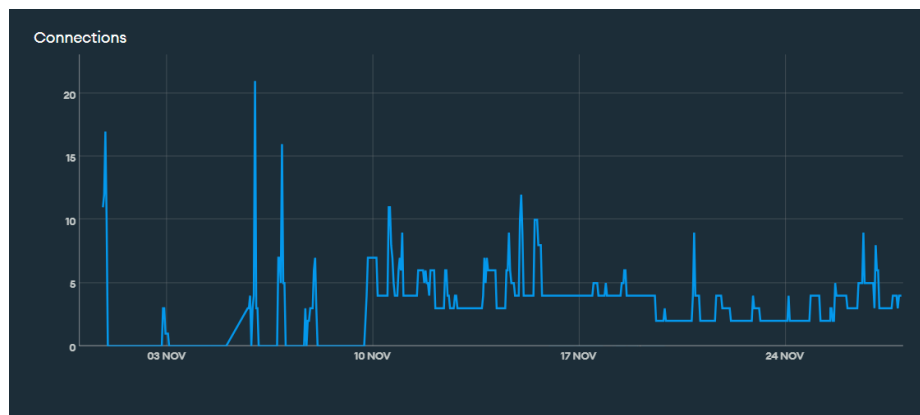


Figura 22. Gráfico de conexiones activas en el cluster

Fuente: Elaboración MongoDB Atlas optainer

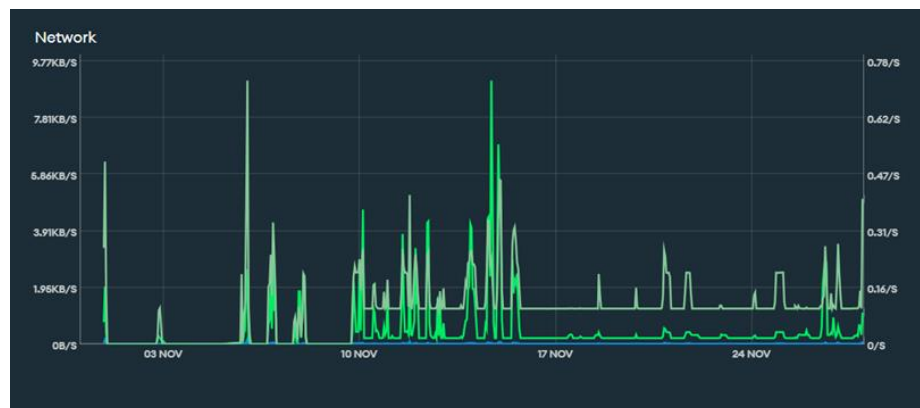


Figura 23. Gráfico de transferencia de datos en la red

Fuente: Elaboración MongoDB Atlas optainer

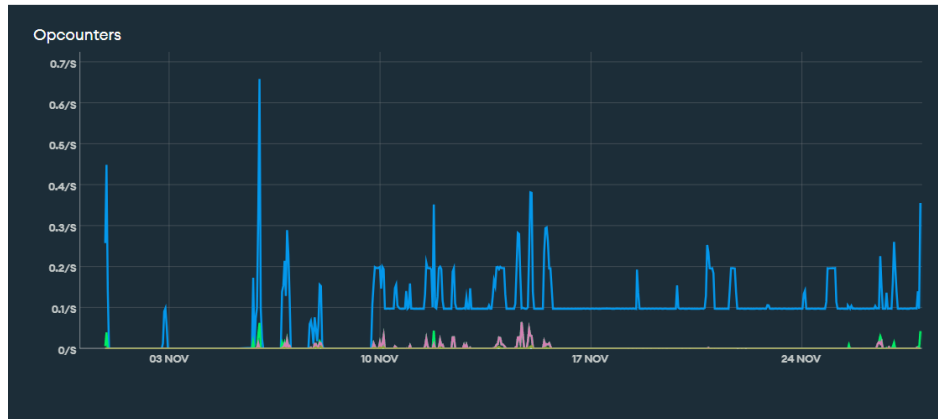


Figura 24. Gráfico de operaciones por segundo (Opcounter)

Fuente: Elaboración MongoDB Atlas optainer

Al contrastar los resultados alcanzados con los objetivos específicos establecidos al comienzo del proyecto, se puede constatar que se han cumplido satisfactoriamente las expectativas de la siguiente manera:

- **Diseño del modelo de datos:** El modelo de datos se desarrolló conforme a las necesidades logísticas de la empresa Aexsa, permitiendo una estructura flexible y escalable que puede ajustarse al crecimiento de la base de datos. Se establecieron relaciones eficientes entre los contenedores y sus ubicaciones, lo que facilita las consultas sobre la ubicación en tiempo real.
- **Implementación en un entorno adecuado:** Se empleó MongoDB, una base de datos NoSQL escalable, que facilita el manejo de grandes volúmenes de datos no estructurados, como las ubicaciones de los contenedores. La implementación en MongoDB Atlas proporcionó una plataforma de base de datos administrada en la nube que facilita la gestión y escalabilidad del sistema sin necesidad de manejar la infraestructura directamente.
- **Consultas y visualización de la información:** Se establecieron las consultas específicas y filtros eficaces para obtener la ubicación de los contenedores. La visualización de los datos se muestra a través del mapa 3D (Figura 14) y la vista de inventario avanzado (Figuras 15, 16 y 17), permitió observar la distribución de los contenedores y mejora a la toma de decisiones por parte de los operadores grúa.
- **Validación del rendimiento:** Las pruebas de rendimiento y las simulaciones de carga corroboraron que la base de datos es capaz de manejar un volumen elevado de contenedores, y que la información se actualiza y consulta de manera eficiente. Se usaron técnicas de optimización de consultas, como el uso de índices y el particionamiento, estás aseguran que el sistema se mantenga ágil incluso en momentos de alta carga de información.

5 Conclusiones

Este proyecto, a través de la creación y puesta en marcha de una base de datos NoSQL vinculada al sistema logístico de contenedores de Aexsa, facilita la administración de datos de ubicaciones de los contenedores en tiempo real y a mejorar la eficiencia en la planificación de movimientos de grúa. Este sistema logra que los operadores de grúas obtengan de forma más rápida los datos requeridos para llevar a cabo sus funciones, disminuyendo considerablemente los periodos de búsqueda de información. Pese a que las pruebas recientes se centraron en confirmar la arquitectura base y las consultas básicas, los resultados evidencian que la base de datos puede gestionar eficazmente el flujo de datos en las condiciones presentes del sistema.

El diseño del modelo de datos basado en las necesidades logísticas de Aexsa, junto con la implementación de consultas optimizadas y una interfaz que permite la visualización en tiempo real, representa un gran avance en la digitalización de procesos logísticos. La capacidad de consultar ubicaciones de contenedores de manera eficiente asegura que la información esté disponible en todo momento, lo que minimiza errores operativos y favorece la toma de decisiones estratégicas.

En cuanto a los detalles adicionales del desarrollo, el trabajo de la base de datos con el Backend tomó aproximadamente ocho semanas. Durante el proceso, fue necesario investigar y aprender sobre el uso de Three.js y sockets para la visualización 3D y actualización de contenedores en tiempo real, lo cual fue un aprendizaje nuevo, ya que no era parte del conocimiento previo. En la formación académica como Ingeniero Informático, las competencias clave incluyeron desarrollo de software, programación, seguridad informática y bases de datos. Estas áreas fueron fundamentales para el diseño del modelo, la optimización de las consultas y la seguridad del sistema, facilitando así la implementación efectiva del sistema.

Aunque el sistema aún no está siendo explotado en su totalidad, debido a que se están llevando a cabo más mejoras y negociaciones con Aexsa, el sistema tiene el potencial de optimizar significativamente los procesos internos y reducir los tiempos operativos. A futuro, al sistema se puede incorporar el respaldo automatizado de datos, índices adicionales para optimizar las consultas, y más particionamiento en caso de que la carga de datos aumente significativamente. Estas mejoras alineadas con las necesidades de escalabilidad del sistema permitirán que 'optainer' se mantenga preparado para manejar volúmenes mayores de datos sin comprometer el rendimiento.

El impacto esperado de este proyecto se enfoca en optimizar los procesos internos y reducir los tiempos operativos, al mismo tiempo que introduce soluciones basadas en tecnologías modernas, aportando un valor significativo a la logística extraportuaria. Este modelo tiene un gran potencial para ser replicado en otras áreas o empresas con necesidades similares, extendiendo así sus beneficios más allá del alcance inicial. Aunque el proyecto se centró en el desarrollo de una base de datos para gestionar la información de los contenedores, abre la puerta a futuras mejoras mediante la integración de herramientas de automatización, como IoT. Estas tecnologías permitirían optimizar aún más el manejo automatizado de las ubicaciones, ampliando las capacidades del sistema y potenciando su impacto en la industria.



Agradecimientos.

Quiero expresar mi más sincero agradecimiento a mi profesor guía, Catherine Gómez, por su invaluable orientación a lo largo de todo este proceso.

También deseo reconocer a mis compañeros de trabajo en Optainer, quienes, con su colaboración, tiempo y conocimientos, hicieron posible este proyecto.

Agradezco profundamente el respaldo constante de mi familia, cuyo apoyo incondicional me brindó la motivación necesaria para culminar este proceso.

De manera especial, agradezco a Xaviera Cabrera, por su constante apoyo emocional y motivación durante el transcurso de esta travesía por la etapa universitaria.

Referencias

- [1] MongoDB. [En línea] <<https://www.mongodb.com/es>>
- [2] React. [En línea]. <<https://es.react.dev/>>
- [3] Node.js. [En línea]. <<https://nodejs.org/en/>>
- [4] Que es Base de Datos. Oracle. [En línea] <<https://www.oracle.com/cl/database/what-is-database/>>
- [5] IoT. [En línea] <<https://www.oracle.com/cl/internet-of-things/>>
- [6] Aexsa. [En línea] <<https://aexsa.cl/>>
- [7] SCRUM. [En línea] <<https://www.scrum.org/resources/what-scrum-module>>
- [8] Mongoose. [En línea] <<https://mongoosejs.com/>>
- [9] Socket.io. [En línea] <<https://socket.io/>>