9-4-2025

CLOUTHESGUARD

BASE DE DATOS MONGO

Integrantes:

Karen Lizbeth Negrete Hernández.

Uriel Abdallah Medina Torres.

Jonathan Baldemar Ramírez Reyes.

Christian Paul Rodríguez Pérez.

Maestro:

M.T.I Norma García Romero.

Materia:

Base de Datos Para Computo en la Nube

1. Base de Datos NoSQL en ClothesGuard

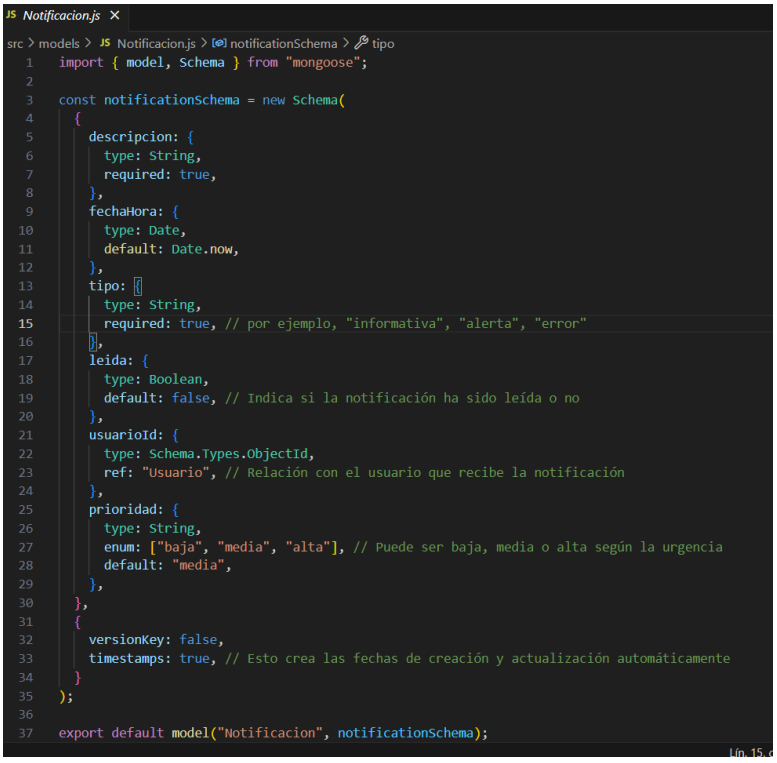
   La solución ClothesGuard (Tendedero Inteligente) se apoya en una arquitectura NoSQL, utilizando MongoDB como sistema de gestión de datos. Esta elección responde a la necesidad de almacenar y consultar información variada y en tiempo real proveniente de:

   - Sensores ambientales y de humedad que detectan condiciones climáticas.
   - Registros de usuarios y sus perfiles, para personalizar alertas y la gestión remota del tendido.
   - Historial de historias o eventos registrados (por ejemplo, comandos de activación o detección de precipitaciones).
   - Notificaciones que alertan a los usuarios sobre cambios en el entorno (como inminente lluvia o estado de secado).

   Esta solución de base de datos es clave para que ClothesGuard pueda adaptarse a un contexto de condiciones climáticas variables y sea escalable a medida que se agreguen nuevas funcionalidades o módulos IoT.

2. Esquema de la Base de Datos

   El diseño se basa en colecciones definidas mediante Mongoose en Node.js, las cuales se integran en el backend para conectar el microcontrolador (ESP32) con las aplicaciones móviles y web. Cada esquema se configura, sin versión, y se usa la flexibilidad de MongoDB para documentos embebidos y referencias.



SensorActuador.js

src > models > SensorActuador.js > sensorSchema > unidad

```
import { model, Schema } from "mongoose";

const sensorSchema = new Schema(
  {
    tipo: {
      type: String,
      required: true,
    },
    nombre: {
      type: String,
      required: true,
    },
    valor: {
      type: Schema.Types.Mixed,
      required: true,
    },
    unidad: {
      type: String,
      default: "",
    },
    accion: {
      type: String,
      default: "",
    },
    fechaHora: {
      type: Date,
      default: Date.now,
    },
  },
  {
    versionKey: false,
    timestamps: true,
  }
);

export default model("SensorActuador", sensorSchema);

```

Story.js

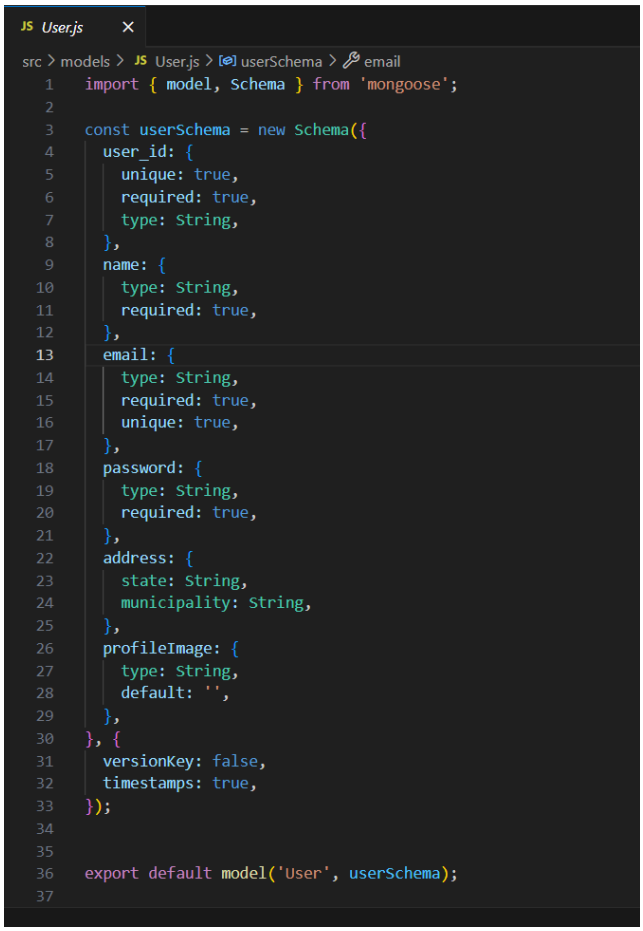
src > models > Story.js > storySchema > diasActivos

```
import { model, Schema } from "mongoose";

const storySchema = new Schema(
  {
    dia: {
      type: Date,
      required: true,
    },
    horasUso: {
      type: String,
      required: true,
    },
    indicaciones: {
      type: String,
    },
    diasActivos: {
      type: Date,
      required: true,
    },
  },
  {
    versionKey: false,
    timestamps: true,
  }
);

export default model("Story", storySchema);

```



3. Explicación de las Entidades y Colecciones
   En el contexto de ClothesGuard, cada entidad definida en la base de datos desempeña un papel fundamental en el funcionamiento del tendedero inteligente. La siguiente descripción profundiza en cada colección, detallando tanto su estructura interna como las consideraciones de rendimiento y relaciones entre datos.

   - User gestiona la identidad y configuración del usuario, lo cual es crítico para la personalización del control remoto del tendido.

   - Story permite llevar un historial de operaciones y eventos que se pueden analizar para optimizar el proceso de secado, identificar patrones de uso y evaluar el rendimiento del sistema en distintos contextos climáticos.

   - SensorActuador es el puente entre el hardware (sensores y actuadores controlados por el ESP32) y el software (aplicaciones web y móviles), garantizando la actualización constante y el registro de datos en tiempo real.

   - Notificación cumple un rol fundamental en la comunicación con el usuario, alertando sobre condiciones ambientales adversas o cambios en el estado del sistema, lo que permite la intervención o ajustes manuales en caso de ser necesario.

3.1. Colección: User

Propósito y Representación:

Esta colección es el almacén central de información de los usuarios finales que utilizarán el sistema ClothesGuard. Permite gestionar no solo las credenciales de acceso, sino también la personalización del servicio, como alertas específicas, configuraciones de perfil y ubicación geográfica (por ejemplo, para adaptar notificaciones en función del clima local).

Campos y Restricciones:

3.1.1.user_id:
- Tipo: String.
- Restricciones: Único y requerido.
- Detalles Adicionales: Este identificador único es utilizado internamente para enlazar la información de un usuario con otros documentos, como notificaciones y registros de actividades, sin exponer datos sensibles.

3.1.2.name:
- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Permite personalizar la interfaz y las comunicaciones del sistema, haciendo que las alertas y mensajes sean más humanizados y dirigidos.

3.1.3.email:
- Tipo: String.
- Restricciones: Requerido y único.
- Detalles Adicionales: El correo electrónico es un método de contacto y, en muchos casos, se utiliza para la verificación de cuentas, recuperación de contraseñas y envío de notificaciones.

3.1.4.password:
- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Las contraseñas se almacenan utilizando algoritmos de hash que garantizan la seguridad de la información, minimizando el riesgo en caso de brechas de datos.

3.1.5.address:
- Tipo: Documento embebido.
- Campos Internos:
  - state: String.
  - municipality: String.
- Detalles Adicionales: Almacenar la dirección completa en un mismo documento permite realizar consultas rápidas basadas en la región o adaptar el servicio según la localidad del usuario (por ejemplo, enviar alertas que consideren condiciones climáticas locales).

3.1.6.profileImage:
- Tipo: String.

- Valor por Defecto: Cadena vacía.
- Detalles Adicionales: Permite almacenar la URL de la imagen de perfil. Esto se integra en la interfaz de usuario para ofrecer una experiencia personalizada y coherente con la identidad del usuario.

Relaciones y Documentos Embebidos:

El uso de documentos embebidos para el campo address simplifica las operaciones sin la necesidad de realizar consultas complejas o múltiples "joins", optimizando así el rendimiento en consultas que involucran la ubicación del usuario.

3.2. Colección: Story

Propósito y Representación:

La colección Story se encarga de almacenar registros relacionados con eventos o actividades diarias que se generan en el sistema. En ClothesGuard, esta colección puede registrar los eventos relacionados con el uso y estado del tendedero inteligente, como los periodos de activación o datos históricos que permitan analizar patrones de uso y rendimiento del dispositivo.

Campos y Restricciones:

3.2.1. dia:

- Tipo: Date.
- Restricciones: Requerido.
- Detalles Adicionales: Representa la fecha en la que se registró la actividad. Es vital para crear cronogramas o para realizar análisis temporales y comparativos entre distintos días.

3.2.2. horasUso:

- Tipo: Int.
- Restricciones: Requerido.
- Detalles Adicionales: Este campo permite almacenar intervalos o rangos de tiempo en que el tendido estuvo activo. Aunque se utiliza un tipo String, podría adaptarse a un formato estandarizado para facilitar la interpretación y los cálculos posteriores.

3.2.3. indicaciones:

- Tipo: String.
- Restricciones: Opcional.
- Detalles Adicionales: Permite agregar observaciones o instrucciones específicas registradas en el momento del evento, lo que puede ser útil para la depuración o para el análisis de incidencias.

3.2.4. diasActivos:

- Tipo: Date
- Restricciones: Requerido.
- Detalles Adicionales: Indica otra fecha relevante –por ejemplo, la fecha de activación o el periodo durante el cual el sistema estuvo funcionando en un modo determinado– lo cual es útil para crear reportes de uso continuo del sistema.

3.3. Colección: SensorActuador

Propósito y Representación:

Esta colección recoge los datos en tiempo real provenientes de los sensores y actuadores instalados en el tendido inteligente. En el contexto de ClothesGuard, se utiliza para registrar datos críticos que permiten la toma de decisiones automáticas, como la activación de mecanismos de seguridad ante la detección de lluvia o la modificación del estado del tendido en función de las condiciones ambientales.

Campos y Restricciones:

3.3.1.tipo:

- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Se utiliza para definir la categoría del dispositivo, por ejemplo, "Sensor" para la detección de clima o "Actuador" para el dispositivo que extiende o retrae el tendido.

3.3.2. nombre:

- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Identifica el componente específico (por ejemplo, “Lluvia”, “Ultrasonico”, “Temperatura” o “Humedad”), lo que permite distinguir entre distintos dispositivos conectados al mismo sistema.

3.3.3. valor:

- Tipo: Schema.Types.Mixed.
- Restricciones: Requerido.
- Detalles Adicionales: Permite almacenar el valor medido en diferentes formatos (numérico o cadena), haciendo flexible el registro de datos que pueden variar según el sensor (por ejemplo, una lectura digital o un valor numérico decimal).

3.3.4. unidad:

- Tipo: String.
- Valor por Defecto: Cadena vacía.
- Detalles Adicionales: Define la unidad de medida (como “cm” para distancias, “°C” para temperaturas o “%” para humedad), lo que garantiza que cada valor registrado sea interpretado correctamente.

3.3.5. accion:

- Tipo: String.
- Valor por Defecto: Cadena vacía.
- Detalles Adicionales: Se utiliza para describir la acción ejecutada en respuesta a un evento del sensor, lo que puede incluir comandos enviados al sistema (por ejemplo, "retraer tendido").

3.3.6. fechaHora:

- Tipo: Date.
- Valor por Defecto: Date.now.

- Detalles Adicionales: Registra la fecha y hora exacta en que se realizó la medición o se ejecutó la acción, lo que es crucial para el análisis en tiempo real y para la generación de informes cronológicos.

3.4. Colección: Notificación

Propósito y Representación:

La colección de Notificación es la encargada de almacenar los avisos y alertas generados por el sistema, dirigidos a los usuarios. En ClothesGuard, estas notificaciones alertan sobre eventos críticos, como la detección de lluvia, el estado de secado de la ropa o problemas técnicos en el sistema.

Campos y Restricciones:

3.4.1. descripcion:
- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Proporciona un texto claro y conciso que explica la razón de la notificación, permitiendo al usuario tomar decisiones informadas (por ejemplo, retirar la ropa o cambiar de modo manual).

3.4.2. fechaHora:
- Tipo: Date.
- Valor por Defecto: Date.now.
- Detalles Adicionales: Registra de forma automática el momento en que se genera la notificación, facilitando el orden y el seguimiento en pantalla.

3.4.3. tipo:
- Tipo: String.
- Restricciones: Requerido.
- Detalles Adicionales: Define la categoría de la notificación (como “informativa”, “alerta” o “error”), lo que permite filtrar y priorizar las notificaciones en la interfaz de usuario.

3.4.4.leida:
- Tipo: Boolean.
- Valor por Defecto: false.
- Detalles Adicionales: Indica si el usuario ha interactuado o visualizado la notificación, permitiendo gestionar estados (por ejemplo, marcando notificaciones pendientes).

3.4.5. usuarioId:
- Tipo: Schema.Types.ObjectId.
- Referencia: Se referencia a la colección User.
- Detalles Adicionales: Establece una relación directa entre la notificación y el usuario destinatario, lo que ayuda a personalizar y segmentar las alertas.

3.4.6. Prioridad
- Tipo: String.
- Restricciones: Enum [“baja”, “media”, “alta”].

- Valor por Defecto: “media”.
- Detalles Adicionales: Permite clasificar la urgencia de las notificaciones, ayudando al sistema y a la interfaz a resaltar aquellas que requieren atención inmediata:

Relaciones y Documentos Embebidos:

El campo usuarioId actúa como un vínculo que une la notificación con el perfil del usuario en la colección User. Esta relación referencial permite mantener la base de datos normalizada, reduciendo la duplicación de información y facilitando la actualización de datos relacionados (por ejemplo, cambios en el nombre del usuario reflejados en diferentes partes del sistema).

4. Integración con la Arquitectura General del Proyecto ClothesGuard
   El diseño de la base de datos se integra en el contexto general del Tendedero Inteligente, tal como se describe en la tesis del proyecto . Entre las consideraciones relevantes se destacan:

   - Interacción con hardware y firmware: El microcontrolador ESP32 registra datos de sensores (DHT11, sensor de lluvia, motores paso a paso) y transfiere la información a través de WebSockets. Estos datos se almacenan en las colecciones de SensorActuador para su posterior análisis y visualización en la aplicación.

   - Conexión con la aplicación móvil y web: Las aplicaciones, desarrolladas en Angular, React Native y Expo, utilizan la base de datos para mostrar información en tiempo real, gestionar comandos remotos y ofrecer al usuario un panel de control completo. La modularidad del backend (estructurado en controladores, modelos y rutas) facilita la integración de la API RESTful que interactúa con MongoDB.

   - Uso en la metodología Scrum: La organización en sprints permite iterar en la optimización del sistema; por ejemplo, ajustes en el registro de datos o mejoras en la consulta del historial en la colección Story se pueden implementar en nuevas versiones sin afectar la operatividad global del sistema.
5. Consideraciones de Seguridad
   La seguridad en ClothesGuard ha sido abordada en distintos niveles:

   Gestión de Datos Sensibles:

   - Contraseñas: Se almacenan de forma segura mediante algoritmos de hash (p.ej., bcrypt), garantizando que la exposición accidental no comprometa la seguridad del usuario.
   - Información personal: Los datos personales se transmiten y almacenan mediante conexiones cifradas (SSL/TLS) y se restringe el acceso tanto en la base de datos como en la aplicación.

   - Acceso restringido a la base de datos: MongoDB se configura utilizando roles internos y autenticación fuerte, además de alojarse en MongoDB Atlas con clusters dedicados para garantizar la alta disponibilidad y protección ante accesos no autorizados.
6. Evaluación del Diseño y su Impacto en el Proyecto ClothesGuard

El diseño del esquema NoSQL cumple con varios objetivos fundamentales para el éxito del Tendedero Inteligente:

Escalabilidad y Flexibilidad:

- La base de datos permite crecimiento horizontal (sharding) y se adapta fácilmente a cambios en el esquema, facilitando la integración de nuevos dispositivos o módulos de información sin interrumpir el servicio.

- La estructura modular y el uso de documentos embebidos en colecciones como User ayudan a optimizar las consultas y reducir la necesidad de operaciones complejas de JOIN, mejorando el rendimiento en un entorno dinámico.

Integración en el Flujo de Trabajo (Metodología Scrum):

- Se permite una rápida incorporación de feedback en cada sprint, lo cual se traduce en ajustes en el almacenamiento y procesamiento de datos en función de las pruebas en campo.

- El enfoque basado en sprints facilita la validación periódica de la funcionalidad, garantizando que los datos registrados (sensores, notificaciones, historias) reflejen con precisión el estado del tendido y las condiciones ambientales.

Rendimiento y Respuesta en Tiempo Real:

- Los índices configurados para campos críticos aseguran una respuesta ágil en las consultas de datos, lo que es esencial para la transmisión en tiempo real a las aplicaciones móviles y web.

- El manejo eficiente de documentos y referencias permite que el sistema responda rápidamente a eventos como la detección de lluvia, activando el mecanismo de protección de forma inmediata.

7. Conclusión

El esquema de la base de datos NoSQL para el proyecto ClothesGuard se ha diseñado considerando tanto la flexibilidad como la eficiencia para integrarse en un sistema complejo de IoT y automatización. El documento detalla de manera integral:

- La estructura y los modelos de datos (User, Story, SensorActuador y Notificación) que permiten almacenar y consultar de forma óptima los registros provenientes de sensores y de la interacción con el usuario.

- La integración del esquema en el contexto global del Tendedero Inteligente, facilitando la comunicación entre hardware (ESP32 y sensores), el backend (API RESTful y Node.js) y las aplicaciones de frontend y móvil.

- Las consideraciones de seguridad, desde la encriptación de contraseñas hasta la restricción de accesos y la configuración de roles en MongoDB.

- La manera en que el diseño soporta la escalabilidad y el rendimiento, permitiendo que el sistema se adapte a futuras demandas y nuevas funcionalidades sin comprometer la operatividad del servicio.

Este enfoque integral y robusto responde a las necesidades del proyecto ClothesGuard, proporcionando una base sólida para el desarrollo y mantenimiento de un tendedero inteligente que ofrezca eficiencia, seguridad y adaptabilidad en entornos climáticos adversos .