Práctico de Desarrollo con Node.js y MongoDB

Objetivo

Desarrollar una aplicación web utilizando Node.js, Express y MongoDB que permita gestionar información de países hispanohablantes, consumiendo datos de una API externa, procesándolos y almacenándolos en una base de datos. Además, crear una interfaz web que muestre y permita interactuar con esta información.

Descripción del Trabajo

1. Consumo de API y Procesamiento de Datos

- **API a consumir**: `https://restcountries.com/v3.1/all`
- **Tarea**:
  - Consumir la API y obtener la lista completa de países.
  - Filtrar los países para **solo** conservar aquellos que tengan, como mínimo, el idioma español (`"spa": "Spanish"`) en su propiedad `languages`. Pueden tener otros idiomas además del español.
  - De cada país seleccionado, eliminar las siguientes propiedades:
    - `translations`
    - `tld`
    - `cca2`
    - `ccn3`
    - `cca3`
    - `cioc`
    - `idd`
    - `altSpellings`
    - `car`
    - `coatOfArms`
    - `postalCode`
    - `demonyms`
  - Agregar una nueva propiedad a cada país:
    - `"creador": "nombreAlumno"` (Reemplazar `"nombreAlumno"` por tu nombre real).

2. Almacenamiento en Base de Datos

- **Base de datos**: MongoDB
- **Tarea**:
    - Almacenar los países procesados en una base de datos MongoDB.
    - Asegurarse de que la estructura de los datos almacenados refleja los cambios realizados (propiedades eliminadas y nueva propiedad añadida).
    - Recomendacion de como debe quedar estructurado cada pais en la base de datos **https://drive.google.com/file/d/171D5b7hXYAKisDpog79IsnJWQVBikZuY/view?usp=sharing**

---

3. Desarrollo de la Interfaz Web

- **Tecnologías**: Node.js, Express, EJS (o el motor de plantillas de tu preferencia).
- **Estructura**:
    - Implementar un **layout base** que incluya una barra de navegación (navbar) y un pie de página (footer) que se mantengan consistentes en todas las vistas.
- **Vista del Dashboard**:
    - Mostrar una tabla con la información de todos los países almacenados.
    - **Campos a mostrar**:
        - `name.spa.official` (Nombre oficial en español)
        - `capital` (Capital)
        - `borders` (Fronteras)
        - `area` (Área)
        - `population` (Población)
        - **Nivel Avanzado** `gini` (Índice Gini)
        - `timezones` (Zonas horarias)
        - `creador` (Nombre del alumno)
    - **Nivel Avanzado** Incluir una fila de **totales** al final de la tabla que sume:
        - La población total (`population`) de todos los países mostrados.
        - El área total (`area`) de todos los países mostrados.
        - El gini promedio (`gini`) de todos los países mostrados.
- **Funcionalidades Adicionales**:
    - **Agregar Nuevo País**:
        - Implementar un formulario para agregar un nuevo país manualmente.
    - **Editar País**:
        - Proporcionar la opción de editar la información de un país existente.
    - **Eliminar País**:
        - Proporcionar la opción de eliminar un país de la base de datos.

---

4. Validaciones

Es **fundamental** que implementes validaciones en tu aplicación para asegurar la integridad de los datos. Las

validaciones deben estar activas y funcionando correctamente en los formularios de agregar y editar.

- **Campos a validar**:
  - `name.official` (Nombre oficial):
    - No debe tener menos de **3 caracteres** ni más de **90 caracteres**.
  - `capital` (Capital):
    - Cada elemento del array debe tener entre **3 y 90 caracteres**.
  - `borders` (Fronteras):
    - Cada código de país en el array debe ser una cadena de **3 letras mayúsculas**.
  - `area` (Área):
    - Debe ser un **número positivo**.
  - `population` (Población):
    - Debe ser un **número entero positivo**.
  - **Nivel Avanzado** `gini` (Índice Gini):
    - Debe ser un número entre **0 y 100**.
- **Manejo de Errores**:
  - Mostrar mensajes claros y amigables al usuario en caso de que alguna validación falle.
  - Evitar que datos inválidos se guarden en la base de datos.

Entregables

1. **Código Fuente**:
   - Proyecto completo con toda la funcionalidad implementada.
   - Instrucciones claras sobre cómo instalar las dependencias y ejecutar la aplicación.
2. **Documentación**:
   - Archivo README que explique:
     - Objetivos del proyecto.
     - Tecnologías utilizadas.
     - Pasos para ejecutar la aplicación.
     - Cualquier consideración especial.
3. **Demostración**:
   - Capturas de pantalla o video que muestren el funcionamiento de la aplicación.
   - Explicación de cómo se implementaron las validaciones y cómo reaccionan ante datos inválidos.

Consideraciones Importantes

- **Claridad y Organización**:
  - Es fundamental que tu código sea claro y esté bien organizado.

    - Utiliza nombres de variables y funciones que reflejen su propósito.
- **Validaciones**:
    - Las validaciones deben estar activas y funcionando correctamente.
    - Prueba tu aplicación ingresando datos inválidos para asegurarte de que las validaciones funcionan.
- **Interfaz de Usuario**:
    - La interfaz debe ser intuitiva y fácil de usar.
    - Asegúrate de que la información se muestra de manera clara.
- **Preguntas y Dudas**:
    - Si algo no está claro en el enunciado, no dudes en consultar.

Pasos Sugeridos para el Desarrollo

1. **Configuración Inicial**:
    - Configura tu entorno de desarrollo.
    - Inicializa un nuevo proyecto Node.js.
    - Instala las dependencias necesarias.
2. **Consumo de la API**:
    - Crea un script que consuma la API y procese los datos según las especificaciones.
    - Asegúrate de que solo se guarden los países que cumplen con el criterio de idioma.
3. **Modelo de Datos**:
    - Define el esquema de Mongoose para el modelo `País`.
    - Incluye solo las propiedades necesarias.
4. **Almacenamiento en MongoDB**:
    - Guarda los países procesados en la base de datos.
    - Verifica que los datos se almacenan correctamente.
5. **Desarrollo del Backend**:
    - Configura Express y las rutas necesarias.
    - Implementa las rutas para listar, agregar, editar y eliminar países.
6. **Implementación de Validaciones**:
    - Añade las validaciones a las rutas de agregar y editar.
    - Asegúrate de manejar los errores de validación correctamente.
7. **Desarrollo del Frontend**:
    - Crea las vistas utilizando el motor de plantillas elegido.
    - Implementa el layout base con navbar y footer.
    - Diseña el dashboard según las especificaciones.

Recursos Adicionales

- **Documentación de Node.js**: https://nodejs.org/es/docs/
- **Express.js**: https://expressjs.com/es/
- **Mongoose**: https://mongoosejs.com/docs/guide.html
- **Express Validator**: https://express-validator.github.io/docs/
- **API Rest Countries**: https://restcountries.com/

Evaluación

Se evaluará:

- **Funcionalidad**:
  - Correcta implementación de las especificaciones.
  - Funcionamiento de las validaciones.
- **Calidad del Código**:
  - Organización y estructura.
  - Legibilidad y comentarios.

¡Buena suerte y éxito en el desarrollo de este proyecto!