¡Se puede utilizar documentación de apoyo oficial de las tecnologías o librerías, no foros, no IA!

1. Contexto

Debes crear una pequeña API REST con Node.js, Express y MongoDB/Mongoose para gestionar personas y cursos.

Hay dos colecciones:

Personas

Cursos

Las relaciones son:

Cada persona está asociada a un curso (por ejemplo, el curso que está tomando o tomó).

Tu objetivo es:

Definir los modelos Mongoose.

Crear los endpoints básicos CRUD (GET, POST, PUT) según el punto 4.

Implementar una búsqueda de información por cédula, usando populate o una agregación (aggregate + $lookup), que devuelva la persona y su curso**.**

2. Requisitos técnicos

Usa Node.js.

Usa Express para la API HTTP.

Usa Mongoose para conectarte a MongoDB y definir los modelos.

Puedes usar .env para la MONGODB_URI (opcional pero recomendable).

Estructura sugerida (no obligatoria):

```
project/
  ├─ src/
  │   ├─ models/
  │   │   ├─ persona.model.js
  │   │   └─ curso.model.js
  │   ├─ controllers/
  │   │   ├─ persona.controller.js
  │   │   └─ curso.controller.js
  │   ├─ routes/
  │   │   ├─ persona.routes.js
  │   │   └─ curso.routes.js
  │   └─ app.js
  └─ server.js
```

Se valorará que el código esté ordenado y modular.

3. Modelos a implementar

3.1. Persona

Colección: personas

Campos mínimos:

nombre (string, requerido)

cedula (string, requerido, único)

email (string, opcional)

curso (ObjectId, referencia al modelo Curso, requerido)

3.2. Curso

Colección: cursos

Campos mínimos:

nombre (string, requerido)

codigo (string, requerido, único)

descripcion (string, opcional)

(Opcional) cualquier otro campo que quieras añadir (duración, nivel, etc.)

4. Endpoints requeridos

Implementa endpoints REST para cada entidad, al menos:

4.1. Personas

GET /personas

Devuelve la lista de personas.

POST /personas

Crea una nueva persona.

GET /personas/:id

Devuelve una persona por su _id.

PUT /personas/:id

Actualiza los datos de una persona.

(Opcional pero valorado: DELETE /personas/:id)

4.2. Cursos

GET /cursos

Devuelve la lista de cursos

POST /cursos

Crea un curso

5. Lógica especial: buscar persona + curso por cédula

Implementa un endpoint para obtener la información de una persona y su curso a partir de la cédula de la persona.

Puedes elegir una de estas formas de URL (elige una y documenta cuál usas):

GET /personas/por-cedula/:cedula

Requisito obligatorio:

La consulta debe usar populate o aggregate:

Opción 1 (populate)

Opción 2 (aggregate + $lookup):

Usar un pipeline de agregación en Persona con $lookup para traer el curso.

La respuesta debe incluir, como mínimo:

Datos de la persona (nombre, cédula, email si aplica).

Datos del curso asociado (nombre, código, etc.).

Ejemplo de formato de respuesta:

```
{
  "persona": {
    "nombre": "Juan Pérez",
    "cedula": "123456789",
    "email": "juan@example.com"
  },
  "curso": {
```

```
  "nombre": "Introducción a Node.js",
  "codigo": "NODE-101",
  "descripcion": "Curso básico de Node.js"
 }
}
```


Si no se encuentra la persona, el endpoint debe responder un mensaje de error claro con un código HTTP apropiado (por ejemplo, 404).

6. Manejo de errores y validaciones mínimas

Valida que los campos requeridos no lleguen vacíos al crear o actualizar.

Maneja errores de Mongoose (por ejemplo, cédula o código duplicado) y devuelve mensajes claros.

Usa try/catch en los controladores para evitar que la API se caiga por errores no controlados.

7. Entregables

Entrega el proyecto en un repositorio (GitHub/GitLab) público, incluyendo:

- Código fuente.
- Archivo README.md con:
  - Ejemplos de requests (JSON de ejemplo para POST/PUT de persona y curso).