TECNM
TECNOLÓGICO NACIONAL DE MÉXICO

ZAMORA

Arquitectura de Servicios

2do Avance del proyecto

Docente: Roberto Suarez Zinzun

Alumno:

Alfaro Herrera Jesús Antonio
Cruz Hernández Juan Manuel
Piñones Ramos Said Rafael
Ramos Arévalos Ricardo

Fecha y Lugar:
14 de marzo de 2025.
Zamora, Mich

INDICE

2do Avance del proyecto

1. Diseño del Modelo de Datos

Para el proyecto Elephocus, se ha optado por utilizar una base de NoSQL, y para ello utilizaremos MongoDB. Esto permitirá una mayor flexibilidad al proyecto, por su capacidad de escalar verticalmente. Como resultado también, de ya tener mejor dominio y concepto de este tipo de base de datos, por su manejo de documentos en formato JSON, además de que el tipo de proyecto está en su mayoría enfocado, para ser una app móvil.

Modelo de Base de Datos (NoSQL - MongoDB)

Para el almacenamiento de datos se han definido las siguientes colecciones:

- Colección de Usuarios (usuarios)

```
{
    "_id": ObjectId(),
    "nombre": String,
    "correo": String,
    "password": String,
    "edad": Number,
    "nivel_academico": String,
    "pais_region": String,
    "estatus": String,
    "tipo_usuario": String
}
```

- Colección de Temas (temas)

```
{
    "_id": ObjectId(),
    "nombre": String,
    "descripcion": String
}
```

- Colección de Flashcards (flashcards)

```
{
    "_id": ObjectId(),
    "pregunta": String,
    "respuesta": String,
    "id_tema": ObjectId(),
```

```
    "autor": ObjectId(("idUsuario"),
    "fecha_creacion": ISODate()
}
```

- Colección de Temas_Flashcards (temas_flashcards)

```
{
    "_id": ObjectId(),
    "id_tema": ObjectId(),
    "id_flashcard": ObjectId()
}
```

2. Script de BD

```
use Elephocus
```

- **Colección “usuarios”**

```
db.createCollection("usuarios");
{
    "_id": ObjectId(),
    "nombre": String,
    "correo": String,
    "password": String,
    "edad": Number,
    "nivel_academico": String,
    "pais_region": String,
    "estatus": String,
    "tipo_usuario": String
}
```

- **Colección “temas”**

```
db.createCollection("temas");
{
    "_id": ObjectId(),
    "nombre": String,
    "descripcion": String
}
```

- **Colección “flashcards”**

```
db.createCollection("flashcards");
{
    "_id": ObjectId(),
    "pregunta": String,
    "respuesta": String,
    "id_tema": ObjectId(),
    "autor": ObjectId("idUsuario"),
    "fecha_creacion": ISODate()
}
```

- **Colección “temas_flashcards”**

```
db.createCollection("temas_flashcards");
{
    "_id": ObjectId(),
    "id_tema": ObjectId(),
    "id_flashcard": ObjectId()
}
```

3. Carga inicial de datos

En este apartado se detalla el proceso de creación, configuración y carga de datos en la base de datos Elephocus utilizando MongoDB. Se establecieron validaciones para garantizar la integridad de los datos y se realizó la inserción de documentos en cada colección.

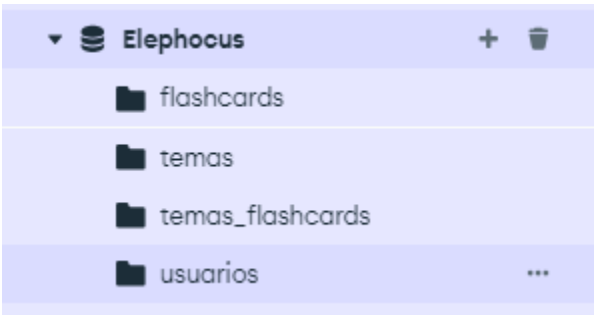
1. Creación de la base de datos

Para iniciar, se creó la base de datos Elephocus en MongoDB Compass o mediante la shell de MongoDB con el siguiente comando:

use Elephocus;

A continuación, se generaron las siguientes colecciones:

- usuarios
- temas
- flashcards
- temas_flashcards



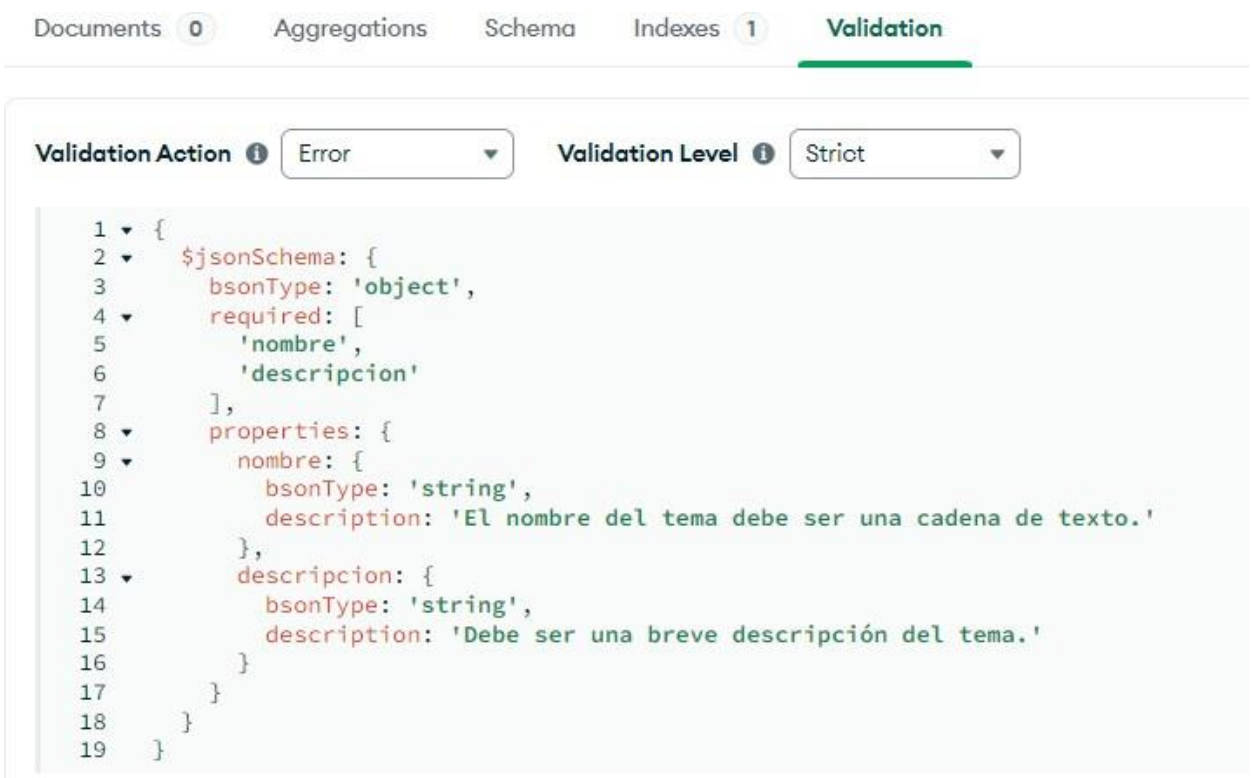
2. Validaciones en cada colección

Para asegurar que los datos ingresados cumplan con una estructura definida, se implementaron validaciones con JSON Schema en MongoDB Compass.

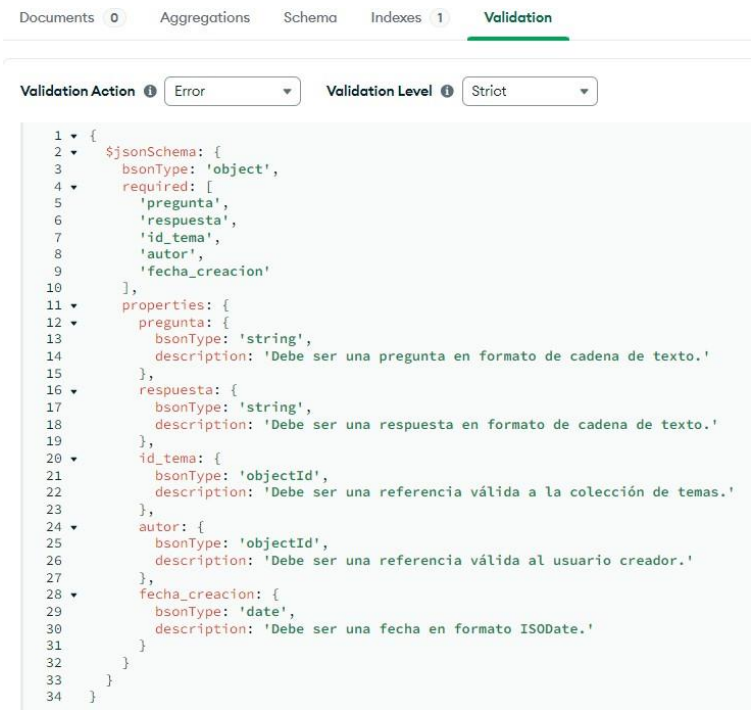
2.1 Validaciones para la colección usuarios


```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'nombre',
      'correo',
      'password',
      'edad',
      'nivel_academico',
      'pais_region',
      'estatus',
      'tipo_usuario'
    ],
    properties: {
      nombre: {
        bsonType: 'string',
        description: 'El nombre del usuario debe ser una cadena de texto.'
      },
      correo: {
        bsonType: 'string',
        pattern: '^\\S+@\\S+\\.\\S+$',
        description: 'Debe ser un correo electrónico válido.'
      },
      password: {
        bsonType: 'string',
        minLength: 8,
        description: 'La contraseña debe tener al menos 8 caracteres.'
      },
      edad: {
        bsonType: 'int',
        minimum: 13,
        description: 'Debe ser un número entero mayor o igual a 13.'
      },
      nivel_academico: {
        bsonType: 'string',
        'enum': [
          'Primaria',
          'Secundaria',
          'Preparatoria',
          'Universidad',
          'Posgrado'
        ],
        description: 'Debe ser un nivel académico válido.'
      },
      pais_region: {
        bsonType: 'string',
        description: 'Debe ser una cadena representando el país o región.'
      },
      estatus: {
        bsonType: 'string',
        'enum': [
          'activo',
          'inactivo'
        ],
        description: 'Debe ser \'activo\' o \'inactivo\'.'
      },
      tipo_usuario: {
        bsonType: 'string',
        'enum': [
          'estudiante',
          'profesor',
          'administrador'
        ],
        description: 'Debe ser un tipo de usuario válido.'
      }
    }
  }
}
```

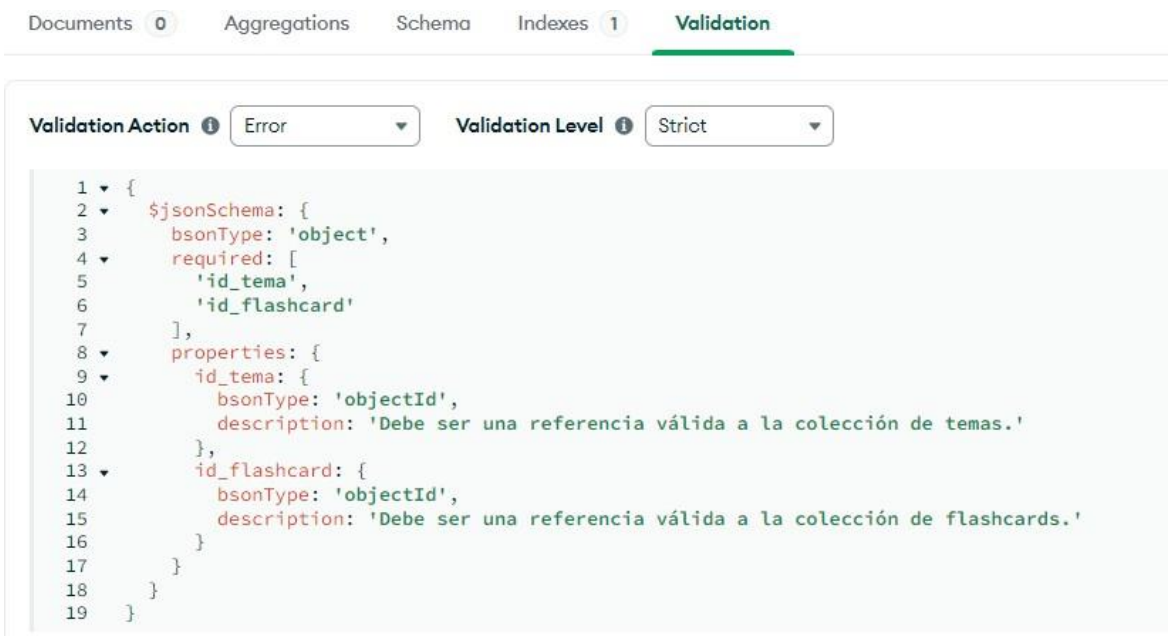
2.2 Validaciones para la colección temas



2.3 Validaciones para la colección flashcards



2.4 Validaciones para la colección temas_flashcards



3. Carga de datos

Para poblar la base de datos, se realizó la inserción de 8 documentos en cada colección.

3.1 Inserción de datos en usuarios

Se insertaron registros con diferentes perfiles, asegurando que cumplan con los requisitos establecidos en las validaciones.

```
> db.usuarios.insertMany([
  {
    _id: ObjectId("640000000000000000000001"),
    nombre: "Juan Pérez",
    correo: "juan.perez@example.com",
    password: "password123",
    edad: 25,
    nivel_academico: "Universidad",
    pais_region: "México",
    estatus: "activo",
    tipo_usuario: "estudiante"
  },
  {
    _id: ObjectId("640000000000000000000002"),
    nombre: "María García",
    correo: "maria.garcia@example.com",
    password: "securepass456",
    edad: 30,
    nivel_academico: "Posgrado",
    pais_region: "Argentina",
    estatus: "activo",
    tipo_usuario: "profesor"
  },
  {
    _id: ObjectId("640000000000000000000003"),
    nombre: "Carlos López",
    correo: "carlos.lopez@example.com",
    password: "mypass789",
    edad: 22,
    nivel_academico: "Preparatoria",
    pais_region: "Chile",
    estatus: "activo",
    tipo_usuario: "estudiante"
  },
  {
    _id: ObjectId("640000000000000000000004"),
    nombre: "Ana Martínez",
    correo: "ana.martinez@example.com",
    password: "anapassword",
    edad: 28,
    nivel_academico: "Universidad",
    pais_region: "Colombia",
    estatus: "activo",
    tipo_usuario: "profesor"
  },
  {
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('640000000000000000000001'),
      '1': ObjectId('640000000000000000000002'),
      '2': ObjectId('640000000000000000000003'),
      '3': ObjectId('640000000000000000000004'),
      '4': ObjectId('640000000000000000000005'),
      '5': ObjectId('640000000000000000000006'),
      '6': ObjectId('640000000000000000000007'),
      '7': ObjectId('640000000000000000000008')
    }
  }
```

3.2 Inserción de datos en temas

Se incluyeron diferentes temas educativos como Matemáticas, Historia, Ciencias, Literatura, entre otros.

```
db.temas.insertMany([
  {
    _id: ObjectId("650000000000000000000001"),
    nombre: "Matemáticas",
    descripcion: "Conceptos básicos y avanzados de matemáticas."
  },
  {
    _id: ObjectId("650000000000000000000002"),
    nombre: "Historia",
    descripcion: "Eventos históricos y contextos importantes."
  },
  {
    _id: ObjectId("650000000000000000000003"),
    nombre: "Ciencias",
    descripcion: "Fundamentos de biología, química y física."
  },
  {
    _id: ObjectId("650000000000000000000004"),
    nombre: "Literatura",
    descripcion: "Análisis de obras literarias y textos."
  },
  {
    _id: ObjectId("650000000000000000000005"),
    nombre: "Geografía",
    descripcion: "Conocimientos sobre países, ciudades y territorios."
  },
  {
    _id: ObjectId("650000000000000000000006"),
    nombre: "Arte",
    descripcion: "Historia del arte y técnicas artísticas."
  },
  {
    _id: ObjectId("650000000000000000000007"),
    nombre: "Programación",
    descripcion: "Conceptos y lenguajes de programación."
  },
  {
    _id: ObjectId("650000000000000000000008"),
    nombre: "Física",
    descripcion: "Leyes y principios de la física clásica y moderna."
  }
]);
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('650000000000000000000001'),
    '1': ObjectId('650000000000000000000002'),
    '2': ObjectId('650000000000000000000003'),
    '3': ObjectId('650000000000000000000004'),
    '4': ObjectId('650000000000000000000005'),
    '5': ObjectId('650000000000000000000006'),
    '6': ObjectId('650000000000000000000007'),
    '7': ObjectId('650000000000000000000008')
  }
}
```

3.3 Inserción de datos en flashcards

Cada flashcard está asociada a un tema y a un usuario creador. Se añadieron preguntas de distintos temas educativos.

```
> db.flashcards.insertMany([
  {
    _id: ObjectId("660000000000000000000001"),
    pregunta: "¿Cuánto es 2+2?",
    respuesta: "4",
    id_tema: ObjectId("650000000000000000000001"), // Matemáticas
    autor: ObjectId("640000000000000000000001"),     // Juan Pérez
    fecha_creacion: ISODate("2025-03-14T10:00:00Z")
  },
  {
    _id: ObjectId("660000000000000000000002"),
    pregunta: "¿Quién fue Simón Bolívar?",
    respuesta: "Líder de la independencia en América Latina.",
    id_tema: ObjectId("650000000000000000000002"), // Historia
    autor: ObjectId("640000000000000000000002"),     // María García
    fecha_creacion: ISODate("2025-03-14T11:00:00Z")
  },
  {
    _id: ObjectId("660000000000000000000003"),
    pregunta: "¿Qué es la fotosíntesis?",
    respuesta: "Proceso mediante el cual las plantas producen energía.",
    id_tema: ObjectId("650000000000000000000003"), // Ciencias
    autor: ObjectId("640000000000000000000003"),     // Carlos López
    fecha_creacion: ISODate("2025-03-14T12:00:00Z")
  },
  {
    _id: ObjectId("660000000000000000000004"),
    pregunta: "¿Quién escribió 'Cien Años de Soledad'?",
    respuesta: "Gabriel García Márquez",
    id_tema: ObjectId("650000000000000000000004"), // Literatura
    autor: ObjectId("640000000000000000000004"),     // Ana Martínez
    fecha_creacion: ISODate("2025-03-14T13:00:00Z")
  },
  {
    _id: ObjectId("660000000000000000000005"),
    pregunta: "¿Cuál es la capital de España?",
    respuesta: "Madrid",
    id_tema: ObjectId("650000000000000000000005"), // Geografía
    autor: ObjectId("640000000000000000000005"),     // Luis González
    fecha_creacion: ISODate("2025-03-14T14:00:00Z")
  },
  {
    _id: ObjectId("660000000000000000000006"),
    pregunta: "¿Qué es el cubismo?",
    respuesta: "Movimiento artístico del siglo XX.",
    id_tema: ObjectId("650000000000000000000006"), // Arte
    autor: ObjectId("640000000000000000000006"),     // Sofía Rodríguez
    fecha_creacion: ISODate("2025-03-14T15:00:00Z")
  },
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('660000000000000000000001'),
      '1': ObjectId('660000000000000000000002'),
      '2': ObjectId('660000000000000000000003'),
      '3': ObjectId('660000000000000000000004'),
      '4': ObjectId('660000000000000000000005'),
      '5': ObjectId('660000000000000000000006'),
      '6': ObjectId('660000000000000000000007'),
      '7': ObjectId('660000000000000000000008')
    }
  }
```

3.4 Inserción de datos en temas_flashcards

Esta colección funciona como una tabla relacional que vincula temas con flashcards específicas.

```
> db.temas_flashcards.insertMany([
    {
      _id: ObjectId("670000000000000000000001"),
      id_tema: ObjectId("650000000000000000000001"),
      id_flashcard: ObjectId("660000000000000000000001")
    },
    {
      _id: ObjectId("670000000000000000000002"),
      id_tema: ObjectId("650000000000000000000002"),
      id_flashcard: ObjectId("660000000000000000000002")
    },
    {
      _id: ObjectId("670000000000000000000003"),
      id_tema: ObjectId("650000000000000000000003"),
      id_flashcard: ObjectId("660000000000000000000003")
    },
    {
      _id: ObjectId("670000000000000000000004"),
      id_tema: ObjectId("650000000000000000000004"),
      id_flashcard: ObjectId("660000000000000000000004")
    },
    {
      _id: ObjectId("670000000000000000000005"),
      id_tema: ObjectId("650000000000000000000005"),
      id_flashcard: ObjectId("660000000000000000000005")
    },
    {
      _id: ObjectId("670000000000000000000006"),
      id_tema: ObjectId("650000000000000000000006"),
      id_flashcard: ObjectId("660000000000000000000006")
    },
    {
      _id: ObjectId("670000000000000000000007"),
      id_tema: ObjectId("650000000000000000000007"),
      id_flashcard: ObjectId("660000000000000000000007")
    },
    {
      _id: ObjectId("670000000000000000000008"),
      id_tema: ObjectId("650000000000000000000008"),
      id_flashcard: ObjectId("660000000000000000000008")
    }
  ]);
```

```
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('670000000000000000000001'),
      '1': ObjectId('670000000000000000000002'),
      '2': ObjectId('670000000000000000000003'),
      '3': ObjectId('670000000000000000000004'),
      '4': ObjectId('670000000000000000000005'),
      '5': ObjectId('670000000000000000000006'),
      '6': ObjectId('670000000000000000000007'),
      '7': ObjectId('670000000000000000000008')
    }
  }
```

4. Definición de los requerimientos de servicios

Servicio de Gestión de Usuarios

Tipo de servicio: Entidad

Responsable: Cruz Hernández Juan Manuel, Piñones Ramos Said Rafael

Operaciones expuestas:

- Crear Usuario
- Obtener Usuarios
- Actualizar Usuario
- Eliminar Usuario

Crear Usuario	
Actor(es)	Usuarios nuevos, Administrador
URL	/usuarios/crear
Método HTTP	POST
Lógica de Negocio	Registrar un nuevo usuario a la base de datos.

Entrada	{
	“nombre”: “string”,
	“email”: “string”,
	“password”: “string”,
	“edad”: “integer”,
	“grado”: “string”
	}

Salida	{
	“_id”:” ObjectId”,
	“nombre”: “string”,
	“email”: “string”,
	“edad”: “integer”,
	“grado”: “string”
	}

Obtener Usuarios	
Actor(es)	Usuarios nuevos, Administrador
URL	/usuarios/(id)
Método HTTP	GET
Lógica de Negocio	Consultar la información de un usuario mediante su _id.

Entrada	{
	“_id”:” ObjectId”
	}

Salida	{ “_id”:” ObjectId”,

	“nombre”: “string”, “email”: “string”, “edad”: “integer”, “grado”: “string” }

Actualizar Usuario	
Actor(es)	Usuarios, Administrador
URL	/usuarios/(id)
Método HTTP	PUT
Lógica de Negocio	Permitir la modificación de los datos del usuario. Se validará que el correo electrónico no este en uso por otro usuario de la aplicación.

Entrada	{
	"nombre": "string",
	"correo": "string",
	"edad": "integer",
	"grado": "string"
	}

Salida	{
	"mensaje": "Usuario actualizado correctamente",
	"_id": "ObjectId",
	"nombre": "string",
	"correo": "string",
	"edad": "integer",
	"grado": "string"
	}

Eliminar Usuario	
Actor(es)	Usuarios, Administrador
URL	/usuarios/(id)
Método HTTP	DELETE
Lógica de Negocio	Eliminar un usuario de la base de datos. Se validará que el usuario exista antes de su eliminación.

Entrada	{
	"_id": "ObjectId"
	}

Salida	{
	"mensaje": "Usuario eliminado correctamente"
	}

Servicio de Gestión de Flashcards

Tipo de servicio: Entidad

Responsables:

Alfaro Herrera Jesús Antonio,

Ramos Arevalos Ricardo

Operaciones expuestas:

- Crear Flashcard
- Obtener Flashcards
- Actualizar Flashcard
- Eliminar Flashcard

Crear Flashcard	
Actor(es)	Usuario registrado
URL	/flashcards/crear
Método HTTP	POST
Lógica de Negocio	Permite a los usuarios crear una nueva flashcard dentro de un temario específico.

Entrada	{
	"id_temario": "ObjectId",
	"pregunta": "string",
	"respuesta": "string"
	}

Salida	{
	"_id": "ObjectId",
	"id_temario": "ObjectId",
	"pregunta": "string",
	"respuesta": "string"
	}

Obtener Flashcards	
Actor(es)	Usuario registrado
URL	/flashcards/(id_temario)
Método HTTP	GET
Lógica de Negocio	Obtiene todas las flashcards asociadas a un temario específico.

Entrada	{
	"id_temario": "ObjectId"
	}

Salida	[
	{
	"_id": "ObjectId",
	"id_temario": "ObjectId",
	"pregunta": "string",
	"respuesta": "string"
	
	}
	]

Actualizar Flashcard	
Actor(es)	Usuario registrado
URL	/flashcards/(id)
Método HTTP	PUT
Lógica de Negocio	Permite modificar el contenido de una flashcard específica.

Entrada	{
	"pregunta": "string",
	"respuesta": "string"
	}

Salida	{
	"mensaje": "Flashcard actualizada correctamente",
	"_id": "ObjectId",
	"pregunta": "string",
	"respuesta": "string"
	}

Eliminar Flashcard	
Actor(es)	Usuario registrado
URL	/flashcards/(id)
Método HTTP	DELETE
Lógica de Negocio	Elimina una flashcard específica. Se validará que la flashcard exista antes de su eliminación.

Entrada	{
	"_id": "ObjectId"
	}

Salida	{
	"mensaje": "Flashcard eliminada correctamente"
	}

Servicio de Gestión de Temarios

Tipo de servicio: Entidad

Responsable: Alfaro Herrera Jesús Antonio

Operaciones expuestas:

- Crear Temario
- Obtener Temarios
- Actualizar Temario
- Eliminar Temario

Crear Temario	
Actor(es)	Usuario registrado
URL	/ temarios /crear
Método HTTP	POST
Lógica de Negocio	Permite a los usuarios crear un nuevo temario.

Entrada	{
	"nombre": "string",
	"descripcion": "string"
	}

Salida	{
	"_id": "ObjectId",
	"nombre": "string",
	"descripcion": "string"
	}

Obtener Temarios	
Actor(es)	Usuario registrado
URL	/ temarios /(id)
Método HTTP	GET
Lógica de Negocio	Obtiene la información de un temario específico.

Entrada	{
	"_id": "ObjectId"
	}

Salida	{
	"_id": "ObjectId",
	"nombre": "string",
	"descripcion": "string"
	}

Actualizar **Temario**	
Actor(es)	Usuario registrado
URL	/ temarios /(id)
Método HTTP	PUT
Lógica de Negocio	Permite modificar la información de un temario.
Entrada	{ "nombre": "string", "descripcion": "string"
	}

Salida	{
	"mensaje": "Temario actualizado correctamente"
	}

Eliminar Temario	
Actor(es)	Usuario registrado
URL	/ temarios /(id)
Método HTTP	DELETE
Lógica de Negocio	Elimina un temario específico.

Entrada	{
	"_id": "ObjectId"
	}

Salida	{
	"mensaje": "Temario eliminado correctamente"
	}

Servicio de Gestión de Categorías

Tipo de servicio: Entidad

Responsable: Piñones Ramos Said Rafael

Operaciones expuestas:

- Crear Categoría
- Obtener Categorías

Crear Categoría	
Actor(es)	Administrador
URL	/categorias/crear
Método HTTP	POST
Lógica de Negocio	Permite registrar una nueva categoría para organizar temarios o flashcards.

Entrada	{
	"nombre": "string",
	"descripcion": "string"
	}

Salida	{
	"_id": "ObjectId",
	"nombre": "string",
	"descripcion": "string"
	}

Obtener Categorías	
Actor(es)	Usuario registrado
URL	/categorias
Método HTTP	GET
Lógica de Negocio	Retorna la lista de categorías disponibles en la aplicación.
Entrada	*NA*

Salida	[
	{
	"_id": "ObjectId",
	"nombre": "string",
	"descripcion": "string"
	}
	]