02 Bases de Datos

Vamos a trabajar con https://mongoosejs.com/

#Instalación Mongoose

```
npm install mongoose --save
```

En
nuestro `app.js` configurar https://mongoosejs.com/docs/connections.html#callback

```
// Conexión base de datos
const mongoose = require('mongoose');

const uri = 'mongodb://localhost:27017/myapp';
const options = {useNewUrlParser: true, useCreateIndex: true};

// Or using promises
mongoose.connect(uri, options).then(
  /** ready to use. The `mongoose.connect()` promise resolves to
mongoose instance. */
  () => { console.log('Conectado a DB') },
  /** handle initial connection error */
  err => { console.log(err) }
);
```

#Schemas

Un Schema nos sirve para estandarizar nuestros documentos en la collection de nuestra base de datos. Te invito a crear otra carpeta con el nombre `models` y dentro un archivo `nota.js` https://mongoosejs.com/docs/guide.html

```
import mongoose from 'mongoose';
const Schema = mongoose.Schema;

const notaSchema = new Schema({
  nombre: {type: String, required: [true, 'Nombre obligatorio']},
  descripcion: String,
  usuarioId: String,
  date:{type: Date, default: Date.now},
  activo: {type: Boolean, default: true}
});
```

```
// Convertir a modelo
const Nota = mongoose.model('Nota', notaSchema);

export default Nota;
```

#Rutas (POST)

Vamos a interactuar con nuestras primeras rutas, para esto dentro de la raíz de tu proyecto crea la carpeta `routes` y luego un archivo `nota.js`

```
import express from 'express';
const router = express.Router();

// importar el modelo nota
import Nota from '../models/nota';

// Agregar una nota
router.post('/nueva-nota', async(req, res) => {
  const body = req.body;
  try {
    const notaDB = await Nota.create(body);
    res.status(200).json(notaDB);
  } catch (error) {
    return res.status(500).json({
      mensaje: 'Ocurrio un error',
      error
    })
  }
});

// Exportamos la configuración de express app
module.exports = router;
```

Configurar `app.js`

```
app.use('/api', require('./routes/nota'));
```

#Postman

https://www.getpostman.com/downloads/ Nos sirve para probar nuestra aplicación, pudiendo enviar peticiones POST, DELETE, GET y PUT según sea necesario.

Levantamos nuestro servidor:

```
npm run devbabel
```

Configuramos a través de POST el llamado a:

```
http://localhost:3000/api/nueva-nota
```

En Headers configuramos:

```
Key: Content-Type
Value: application/x-www-form-urlencoded
```

Y en body seleccionamos `x-www-form-urlencoded` y agregamos los campos necesarios para agregar una nueva categoria:

```
nombre | Categoria
descripcion | Descripción Categoria
usuarioId | kdkdkdk
```

Damos clic en Send y cruzamos los dedos para ver si todo funciona ok.


```
{
    "_id": "5d6c830664b3a8144c5ad6f1",
    "nombre": "Categoria",
    "descripcion": "Descripción Categoria",
    "usuarioId": "kdkdkdk",
    "activo": true,
    "date": "2019-09-02T02:48:38.281Z",
    "__v": 0
}
```


#Rutas GET

```
// Get con parámetros
router.get('/nota/:id', async(req, res) => {
  const _id = req.params.id;
  try {
    const notaDB = await Nota.findOne({_id});
    res.json(notaDB);
  } catch (error) {
    return res.status(400).json({
      mensaje: 'Ocurrio un error',
      error
    })
```

```
  }
});
```

```
// Get con todos los documentos
router.get('/nota', async(req, res) => {
  try {
    const notaDb = await Nota.find();
    res.json(notaDb);
  } catch (error) {
    return res.status(400).json({
      mensaje: 'Ocurrio un error',
      error
    })
  }
});
```

#Rutas DELETE

```
// Delete eliminar una nota
router.delete('/nota/:id', async(req, res) => {
  const _id = req.params.id;
  try {
    const notaDb = await Nota.findByIdAndDelete({_id});
    if(!notaDb){
      return res.status(400).json({
        mensaje: 'No se encontró el id indicado',
        error
      })
    }
    res.json(notaDb);
  } catch (error) {
    return res.status(400).json({
      mensaje: 'Ocurrio un error',
      error
    })
  }
});
```

#Rutas PUT

```
// Put actualizar una nota
router.put('/nota/:id', async(req, res) => {
  const _id = req.params.id;
  const body = req.body;
  try {
    const notaDb = await Nota.findByIdAndUpdate(
```

```
      _id,
      body,
      {new: true});
    res.json(notaDb);
  } catch (error) {
    return res.status(400).json({
      mensaje: 'Ocurrio un error',
      error
    })
  }
});
```