CRUD MONGODB MACOS

Paso 1: Instalar MongoDB

1. **Instalar Homebrew (si no lo tienes):**
   Abre la terminal y ejecuta el siguiente comando para instalar Homebrew:

   ```
   /bin/bash -c "$(curl -fsSL
   https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
   ```

2. **Instalar MongoDB:**
   Una vez que Homebrew esté instalado, instala MongoDB con el siguiente comando:

   ```
   brew tap mongodb/brew
   brew install mongodb-community
   ```

3. **Iniciar MongoDB:**
   Inicia el servicio de MongoDB:

   ```
   brew services start mongodb/brew/mongodb-community
   ```

Paso 2: Configurar el Entorno de Desarrollo

1. **Instalar Node.js (si no lo tienes):**
   Para crear una aplicación, es útil tener Node.js instalado. Instálalo con Homebrew:

   ```
   brew install node
   ```

2. **Instalar un editor de código:**
   Puedes usar Visual Studio Code, que es popular y gratuito. Descárgalo desde Visual Studio Code.

Paso 3: Crear la Aplicación

1. **Inicializar un nuevo proyecto Node.js:**
   Crea una nueva carpeta para tu proyecto y navega a ella en la terminal:

   ```
   mkdir inventario-tienda
   cd inventario-tienda
   ```

   Inicializa un nuevo proyecto Node.js:

   ```
   npm init -y
   ```

2. **Instalar dependencias:**
   Instala las dependencias necesarias, incluyendo `express` para el servidor y `mongodb` para la base de datos:

```
npm install express mongodb
```

Paso 4: Escribir el Código

1. **Crear el archivo principal (`index.js`):**
   En la carpeta del proyecto, crea un archivo `index.js` y añade el siguiente código básico para conectarse a MongoDB y configurar Express:

```
const express = require("express");
const { MongoClient, ObjectId } = require("mongodb");
const app = express();
const port = 3000;

// Middleware para parsear JSON

app.use(express.json());

// Conexión a MongoDB

const url = "mongodb://localhost:27017";
const client = new MongoClient(url);

let db;
async function connectDB() {
  try {
    await client.connect();

    db = client.db("tienda");

    console.log("Conectado a MongoDB");
  } catch (err) {
    console.error("Error conectando a MongoDB:", err);
  }
}

connectDB();
```

2. **Crear el esquema de la base de datos:**
   Diseña tu esquema de base de datos y añade algunas rutas para gestionar el inventario. Aquí hay un ejemplo básico:

```
// Rutas de la aplicación
app.get("/", (req, res) => {
  res.send("Gestión de Inventario de Tienda en Línea");
});

// Añadir un producto
app.post("/productos", async (req, res) => {
  const producto = req.body;
  try {
    const result = await db.collection("productos").insertOne(producto);
    res.send(result);
  } catch (err) {
    res.status(500).send(err);
  }
});
```

```
// Obtener todos los productos
app.get("/productos", async (req, res) => {
  try {
    const items = await db.collection("productos").find().toArray();
    res.send(items);
  } catch (err) {
    res.status(500).send(err);
  }
});

// Obtener un producto por su ID
app.get("/productos/:id", async (req, res) => {
  const id = req.params.id;
  try {
    const producto = await db
      .collection("productos")
      .findOne({ _id: new ObjectId(id) });
    if (producto) {
      res.send(producto);
    } else {
      res.status(404).send({ message: "Producto no encontrado" });
    }
  } catch (err) {
    res.status(500).send(err);
  }
});

// Actualizar un producto
app.put("/productos/:id", async (req, res) => {
  const id = req.params.id;
  const nuevoProducto = req.body;
  try {
    const result = await db
      .collection("productos")
      .updateOne({ _id: new ObjectId(id) }, { $set: nuevoProducto });
    if (result.matchedCount > 0) {
      res.send({ message: "Producto actualizado exitosamente" });
    } else {
      res.status(404).send({ message: "Producto no encontrado" });
    }
  } catch (err) {
    res.status(500).send(err);
  }
});

// Eliminar un producto
app.delete("/productos/:id", async (req, res) => {
  const id = req.params.id;
  try {
    const result = await db
      .collection("productos")
      .deleteOne({ _id: new ObjectId(id) });
    if (result.deletedCount > 0) {
      res.send({ message: "Producto eliminado exitosamente" });
    } else {
      res.status(404).send({ message: "Producto no encontrado" });
    }
```

```
  } catch (err) {
    res.status(500).send(err);
  }
});
```

3. Creamos la función para conectarnos al servicio de express, con la finalidad de que nuestra api escuche desde el puerto y ruta de conexión.

```
app.listen(port, () => {
console.log(`Servidor corriendo en http://localhost:${port}`);
});
```

Paso 5: Probar la Aplicación

1. **Ejecutar el servidor:**
   En la terminal, ejecuta el siguiente comando para iniciar tu aplicación:

   ```
   node index.js
   ```

Paso 6: Resumen de Endpoints y Ejemplos

1. **Añadir un producto (`POST`):**
   - **URL:** `http://localhost:3000/productos`
   - **Método:** `POST`
   - **Body (JSON):** json Copiar código

```
{
  "nombre": "Camiseta",
  "descripcion": "Camiseta de algodón de alta calidad",
  "precio": 19.99,
  "stock": 50,
  "categoria": "Ropa",
  "proveedor": {
    "nombre": "Proveedor Ejemplo",
    "contacto": "contacto@proveedorejemplo.com"
  }
}
```

2. **Obtener todos los productos (`GET`):**
   - **URL:** `http://localhost:3000/productos`
   - **Método:** `GET`
3. **Obtener un producto por ID (`GET`):**
   - **URL:** `http://localhost:3000/productos/60d5ec49f8d4c13e0c4d32d9`
   - **Método:** `GET`
4. **Actualizar un producto (`PUT`):**
   - **URL:** `http://localhost:3000/productos/60d5ec49f8d4c13e0c4d32d9`
   - **Método:** `PUT`
   - **Body (JSON):** json Copiar código

```
{
  "nombre": "Camiseta Actualizada",
  "descripcion": "Camiseta de algodón actualizada de alta calidad",
  "precio": 25.99,
  "stock": 45,
  "categoria": "Ropa",
  "proveedor": {
    "nombre": "Proveedor Actualizado",
    "contacto": "nuevo_contacto@proveedorejemplo.com"
  }
}
```

5. **Eliminar un producto (`DELETE`):**
   - **URL:** `http://localhost:3000/productos/60d5ec49f8d4c13e0c4d32d9`
   - **Método:** `DELETE`
6. **Probar las rutas:**
   Puedes utilizar herramientas como Postman o Insomnia para probar las diferentes rutas (`POST`, `GET`, `PUT`, `DELETE`) de tu API.

Recursos Adicionales

- Documentación de MongoDB
- Guía de Express.js
- Node.js Documentation