

**Centro de ciencias básicas**

**DEPARTAMENTO DE SISTEMAS DE INFORMACIÓN**

**CURSO:** ADMINISTRACION Y DISEÑO WEB

**Manual Técnico**

**MAESTRA:**

Margarita Mondragón Arellano

**ALUMN@(S):**

Sixto Jesus Escobar Cervantes-293555

Cesar Eduardo López Baltazar- 291140

Alan Ellud Gonzales Frausto- 298134

Jesus Aramis Acevedo Rivera- 292047

AGUASCALIENTES, AGS., 01 DE DICIEMBRE DE 2023

**Manual Técnico del API**

En el parte de controllers se agregó el siguiente código

const productoCtrl = {};

const Producto = require('../models/Producto');

En esta parte se hace el control para consultar todos los documentos

productoCtrl.getProductos = async (req, res) =>

{

    Aquí se consultan los documentos

    const ListaProductos = await Producto.find();

    Y con esta línea se envían los documentos

    res.json(ListaProductos);

}

Después de controllers nos vamos al código de models

Aquí se importan las clases Schema y model

const { Schema, model } = require('mongoose');

const productoSchema = new Schema({

En el siguiente código se crean los campos para ingresar el código del producto(que podría ser su id), el nombre, la descripción, el precio, y un campo para poderle poner imagen a ese producto.

    Codigo: {

        type: Number,

        unique: true,

        required: true

    },

    Nombre: {

        type: String,

        required: true

    },

    Descripcion: String,

    Precio: {

        type: Number,

        required: true

    },

    ImgURL: String

});

En esta parte se crea un modelo llamado “Producto” utilizando el esquema que se creó anteriormente y se exporta ese modelo para que pueda ser utilizado en otros archivos.

const Producto = model('Producto', productoSchema);

module.exports = Producto;

En el aparato de los routes

En esta parte son las rutas de productos del código de models

const { Router } = require('express');

const router = Router();

Aquí se obtienen los controladores del código de controllers

const { getProductos, getProducto, createProducto, updateProducto, deleteProducto } = require('../controllers/producto.controller');

Y estas son las rutas para métodos que no requieren ID como podría ser la consulta de todos los documentos o las rutas de insertar un documento.

router.route('/')

    .get(getProductos)

    .post(createProducto);

Aquí están las rutas para métodos que si requieren ID que son las consultas del documento, actualizar el documento o eliminar un documento.

router.route('/:codigo')

    .get(getProducto)

    .put(updateProducto)

    .delete(deleteProducto);

module.exports = router;

Ahora para la parte del app.js se conto con el siguiente codigo

Esta es la dependencia para cargar las variables del .env

require('dotenv').config();

Y aquí se encuentra la dependencia de las rutas, del express y del cors

const path = require('path');

const express = require('express');

const cors = require('cors');

const app = express();

En esta parte son los ajustes que es a donde se le asigna un puerto que nosotros elegimos el puerto 4000

app.set('port', process.env.PORT || 4000);

app.use(cors());

app.use(express.json());

app.use(express.static(path.join(\_\_dirname, '1\_FrontEnd')));

Aquí se referencia como ruta al router

app.use('/producto', require('./routes/producto'));

module.exports = app;

En este código es el database.js que se hizo lo siguiente

const mongoose = require('mongoose');

Esta línea carga las variables de entorno desde el archivo. env

require('dotenv').config();

Esta parte se encarga de obtiene la URI del archivo. env

const URI = process.env.MONGO\_URI;

Se conecta a la base de datos

mongoose.connect(URI);

const connection = mongoose.connection;

Ya cuando la conexión sea correcta se manda un mensaje para así saber que la base de datos ya está conectada.

connection.once('open', () => {

    console.log('Base de datos conectada')

});

Y esto es lo que sería el index.js

const app = require('./app');

Se importa la conexión a la base de datos

require('./database');

async function main()

{

    await app.listen(app.get('port'));

    console.log('Server on port ', app.get('port'));

}

Llama a la función main para iniciar la aplicacion

main();

Y el código del .env, esto básicamente lo que hace es la conexión con la base de datos, aquí se indica el puerto de salida y se agrega la contraseña y usuario para que esta pueda funcionar.

PORT=4000

MONGO\_URI=mongodb+srv://DataBaseAdminCesar:AdminMacizo@cluster0.jehx0h1.mongodb.net/TiendaConsolas?retryWrites=true&w=majority

**MODULOS NECESARIOS**

Los módulos que usamos en este caso fueron mongo y express, estos ya que fueron algunos vistos durante el semestre y con los que a lo largo de este fuimos trabajando.

Para la instalación del mongodb, nos fuimos directamente a la página en línea que tienen en la que solamente nos pide registrarnos para poder iniciar a trabajar con ella. Una vez ya registrados nos saldría una página como esta

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Y aquí ya podemos iniciar a crear las tablas necesarias.

Al momento de crear nosotros unas tablas nos saldría algo así

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En este caso agarre unas tablas solo como ejemplo ya que nuestras tablas y demás ya están creadas.

Ahora, en la instalación de express, esta la hicimos mediante el cmd de Windows y para ello se necesitan ciertos pasos como los siguientes:

Primero abrimos el cmd en el directorio donde queramos crear el proyecto

Texto

Descripción generada automáticamente

Se pone el comando npm init -y para para iniciar un nuevo proyecto de Node.js y nos saldrá algo así

Texto

Descripción generada automáticamente

Después de esto agregamos el comando npm install express y esperamos a que se instale

Texto

Descripción generada automáticamente

Y ya con esto podemos crear el archivo app.js como ejemplo para configurar nuestro express.

Una vez que lo hayamos configurado ejecutamos el siguiente comando



Y nuestra aplicación estará disponible en el localhost con el puerto que pusimos en la configuración de nuestro express