07-restserver

**Objetivos:**

* **Configuraciones iniciales para poder crear el REST server**
* **Instalacion de la base de datos Mongo DB**
* **Crear un EXPRESS REST server**
* **Crear la configuracion global de la aplicacion**

**CLASE 77 . Instalacion de MongDB**

Objetivo: Instalacion de MongoDB – base de datos no relacional

En navegador web

* Buscar y abrir pagina ‘mongodb.com’
* Click a boton verde ‘Try free’

Entra a pantalla ‘MongoDB Download Center’

* Ubicar menu horizontal a media pagina
* Click a opcion ‘Server’
* Entra a pantalla ‘MongoD Community Server’

Se muestra la version, el sistema operativo ‘Windows’ y el package ‘MSI’

* Clck en boton verde ‘Download’ para instalar y ‘next’ a todas las demas opciones
* En menu vertical de la derecha click a ‘Installation instructions’

Entra a pantalla de instrucciones de ‘Install MongoDB Community Edition’

* Localizar seccion

Run MongoDB Community Edition from the Command Interpreter

* Ubicar la seccion

### Create database directory

* Seleccionar las siguientes instrucciones, copiarlas y ejecutarlas en consola

**cd** C:\

**md** "\data\db"

* Ubicar la seccion

### Start your MongoDB database

* Seleccionar la siguiente instruccion, copiarla y ejecutarla en consola

"C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe" --dbpath="c:\data\db"

Esta instrucción ‘levanta/arranca’ el servicio de MongoDB; Importante: Deben ponerse las comillas al inicio y al final de la instrucción;

al ejecutarla se muestran una lista de mensajes relacionados con el arranque, en la segunda linea se muestra ‘port 27017’; este puerto sera utilizado despues

al final aparece el mensaje

[initandlisten] waiting **for** connections on port 27017

Que indica que esta corriendo el servicio de MongoDB

* Cuando se quiera ‘bajar/cancelar’ el servicio de MongoDB teclear ‘Cntrl-C’

**CLASE 78 . Robo 3T – Herramienta para manejar Mongo**

Objetivo: Instalacion de Robo 3T y creacion de primera base de datos

En navegador web

* Buscar pagina ‘Robo 3T – formerly Robomongo’ (robomongo.org)
* Click a boton verde ‘download Robo 3T’

Pasa a sig pantalla

* Click a boton verde ‘download Robo 3T’
* Seleccionar sistema operativo
* Click en version correspondiente
* Click a todos los NEXT siguientes

En consola

* Teclear:

Cd 07-restserver

* Teclear:

"C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe" --dbpath="c:\data\db"

Esta instrucción Levanta la base de datos ‘Mongo’

En escritorio

* Abrir app ‘robo 3T’ (con el icono creado cuando se instalo)

En app ‘Robo 3T’

Se abre pagina principal

Se abre ventana ‘MongoDB Connections’

* Click a opcion ‘Create’ en menu horizontal superior

Se abre ventana ‘Connection Settings’

* Parametros:

Type: Direct connection

Name: Localhost

Address: Localhost

Puerto: 27017 (es el que se muestra cuando se levanta’Mongo’

* Click a boton ‘save’

Regresa a ventana ‘MongoDB Connections’

Se muestra la conexión creada ‘Localhost’

* Seleccionar conexión ‘localhost’
* Click a boton ‘connect’ en la parte inferior de la ventana

Pasa a pantalla principal ‘Robo 3T’

* En menu vertical lado derecho, click boton derecho en item ‘Localhost (n)’

Se abre menu qry

* Click a opcion ‘Create database’

Se abre ventana ‘database name’

* Teclear database name: café

Se muestra esa nueva base de datos en el menu vertical a la derecha

* Abrir descendencia de base de datos ‘café’

Se muestran folder descendientes

* Click boton derecho en folder ‘collections (n)’

Se abre menu qry vertical

* Click a opcion ‘Create collection’
* Collection name: ususarios
* Click a boton ‘create’
* Se muestra la coleccion ‘ususarios’ bajo el folder ‘collections(n)’
* Doble click a la colección ‘usuarios’

Se muestra ventana ‘db.getcollection(‘usuarios’).find({}) sin ningun registro

* Click boton derecho en colección ‘usuarios’

Se abre menu qry

* Click a opcion ‘Insert Document’

Se abre pantalla ‘Insert Document’

* Teclear entre los corchetes ‘ “nombre”:”Fernando”
* Click a boton ‘save’
* Doble click a la colección ‘usuarios’

Se muestra ventana ‘db.getcollection(‘usuarios’).find({}) con este primer registro creado

* Abrir este primer registro
* Se muestran dos campos, el primero es un ‘\_id’ creado automaticamente por ‘mongo’ y el segundo es el que se inserto recientemente

**CLASE 79 . iniciando el proyecto - RESTServer**

Objetivo: crear un programa ‘server.js’ que despliegue el texto ‘Hello World’ usando ‘express REST’

En explorador de archivos

* Si no existe, crear el folder ‘07-restserver’
* Arrastrarlo a Visual Studio Code para que lo incorpore

En consola

* Teclear

Cd C:\Users\EG\_Lobo\Documents\Mis documentos\aa Major 1 EGB\EGB EDU\TAKEN\NODE\07-restserver

* Teclear

‘Npm init’

Responder si a todas los prompts

Esta instrucción inicializa ‘npm’ en el folder y crea los archivos ‘.json’ para asi poder realizar instalaciones y todo los demas servicios necesarios

En navegador web

* Buscar ‘npm express’
* Abrir pagina ‘Express -npm’ - <https://www.npmjs.com/package/express>
* Seleccionar opcion ‘npm i express’ y copiarla a consola

Esta instrucción es para instalar express

En consola

* Pegar instrucción ‘npm i express’ y teclear al final ‘—save’ (dos guiones) y ejecutarla

Se instalara express

En visual studio code

* En folder ‘07-restserver’ crear folder ‘server’
* En folder ‘server’ crear archivo ‘server.js’

En navegador web

En pagina ‘npm express’

* Seleccionar estas lineas

const express **=** require('express')

const app **=** express()

app.get('/', function (req, res) {

  res.send('Hello World')

})

app.listen(3000)

* Pegarlas en Visual Studio Code en archivo ‘server.js’

En visual studio code

* Modificar esta linea

app.listen(3000, () => {

console.log('Escuchando puerto: ', 3000);

});

En consola

* Teclear: ‘Node server/server’
* Se desplegara el mensaje ‘Escuchando puerto: 3000’

En postman

En prompt para URL

* Checar que sea peticion ‘GET’
* teclear ‘localhost:3000’
* Click a boton ‘SEND’

En ventana inferior

* En opcion ‘Body’ debe aparecer el texto ‘Hello World’
* En opcion ‘Headers’, en linea ‘Content type’ debe decir ‘text/html’

En visual studio code

* Modificar esta linea

res.json('Hello World')

En postman

En prompt para URL

* Checar que sea peticion ‘GET’
* teclear ‘localhost:3000’
* Click a boton ‘SEND’

En ventana inferior

* En opcion ‘Body’ debe aparecer el texto ‘Hello World’
* En opcion ‘Headers’, en linea ‘Content type’ sigue mostrando ‘text/html’ (la razon es porque no se ha detectado que se hizo un cambio en el codigo porque el servidor fue ‘levantado’ con el comando ‘node’

En consola

* Bajar el servidor con ‘cntrl-C’
* Teclear ‘nodemon server/server’

Con esta instrucción se detectara cualquier cambio que se haga en el archivo server

Se despliegan varias lineas que indcan que se esta monitoreando el archivo server.js

En postman

En prompt para URL

* Checar que sea peticion ‘GET’
* teclear ‘localhost:3000’
* Click a boton ‘SEND’

En ventana inferior

* En opcion ‘Body’ debe aparecer el texto ‘Hello World’
* En opcion ‘Headers’, en linea ‘Content type’ debe decir ‘application/json’

**CLASE 80 . Peticiones HTTP – GET – PUT – POST - DELETE**

Objetivo:

* Modificar el programa ‘server.js’ para que haga peticiones ‘get’, ‘put’, ‘post’, ‘delete’;
* En el ‘post’ que obtenga datos del ‘body’ de postman
* En el ‘put’ que obtenga datos del URL despues del ‘/usuario/’

LEVANTAR SERVIDOR

En consola

* Teclear

Cd C:\Users\EG\_Lobo\Documents\Mis documentos\aa Major 1 EGB\EGB EDU\TAKEN\NODE\07-restserver

* Teclear: ‘nodemon server/server’
* Se desplegaran varias lineas
* Se desplegara el mensaje ‘Escuchando puerto: 3000’

En Visual Studio Code

En file server.js

* Modificar lineas para que quede como se indica la peticion ‘get’

app.get('/usuario', function(req, res) {

res.json('get usuario')

})

En postman

* En recuadro URL

GET

<http://localhost:3000/usuario>

SEND

* En ventana abajo se despliega

‘get usuario’

En Visual Studio Code

En file server.js

* Poner estas lineas debajo de la peticion ‘get’

app.post('/usuario', function(req, res) {

res.json('post usuario')

})

En postman

* En recuadro URL

POST

<http://localhost:3000/usuario>

SEND

* En ventana abajo se despliega

‘post usuario’

En Visual Studio Code

En file server.js

* Poner estas lineas debajo de la peticion ‘post’

app.put('/usuario', function(req, res) {

res.json('put usuario')

})

app.delete('/usuario', function(req, res) {

res.json('delete usuario')

})

En postman

* En recuadro URL

PUT

<http://localhost:3000/usuario>

SEND

* En ventana abajo se despliega

‘put usuario’

* En recuadro URL

DELETE

<http://localhost:3000/usuario>

SEND

* En ventana abajo se despliega

‘delete usuario’

* En recuadro URL

PUT

http://localhost:3000/usuario/81379182

SEND

* En ventana abajo se despliega

‘Cannot PUT /usuario/8137982’ ya que no se ha programado para que reciba un dato despues de ‘/usuario/’

PROGRAMAR PARA RECIBIR DATO DESPUES DE ‘/USUARIO/’

En Visual Studio Code

En file server.js

* Modificar lineas en la peticion ‘put’ para que quede asi

app.put('/usuario/:id', function(req, res) {

let id = req.params.id;

res.json({

id

});

});

En postman

* En recuadro URL

PUT

http://localhost:3000/usuario/81379182

SEND

* En ventana abajo se despliega

{

"id": "81379182"

}

SIMULAR QUE UNA APLICACIÓN LE MANDA DATOS AL SERVIDOR

En postman

en menu horizontal segunda linea,

* click a opcion ‘Body’
* click a boton-radio ‘x-www-form-urlencoded’

en recuadros debajo de columnas: ‘key’ y ‘value’ teclear

nombre | ernesto

edad | 59

mail | eglobo911@gmail.com

challenge: programar ‘server.js’ para obtener estos datos que envia la aplicacion

PROGRAMAR server.js PARA LEER ESOS DATOS DEL SERVIDOR

En navegador web

* teclear ‘npm bodyparser’
* click a opcion ‘body-parser npm’

se abre pagina ‘body-parser’

* seleccionar la instrucción

npm i body-parser

* copiarla a consola y teclear al final ‘- - save’

ver que se instale sin error

en pagina ‘npm-body-parser’

* localizar y seleccionar linea

var bodyParser **=** require('body-parser')

* copiarla a file ‘server.js’ donde se indica

const app = express()

const bodyParser = require('body-parser')

* localizar y seleccionar lineas

*// parse application/x-www-form-urlencoded*

app.use(bodyParser.urlencoded({ extended**:** false }))

*// parse application/json*

app.use(bodyParser.json())

* copiarlas a ‘server.js’ donde se indica

const bodyParser = require('body-parser')

*// parse application/x-www-form-urlencoded*

app.use(bodyParser.urlencoded({ extended: false }))

*// parse application/json*

app.use(bodyParser.json())

en visual studio

* modificar la peticion ‘post’ para que quede asi

app.post('/usuario', function(req, res) {

let body = req.body;

res.json({

persona: body

});

});

En postman

* En recuadro URL

POST

http://localhost:3000/usuario

SEND

* En ventana abajo se despliega

{

"persona": {

"nombre": "Ernesto",

"edad": "59",

"correo": "eglobo911@gmail.com"

}

}

**CLASE 81 . Codigos de respuestas HTTP**

Objetivo: descargar pdf con los codigos



**CLASE 82 . Usando codigos de respuesta HTTP en Express**

Objetivo: usar en express los codigos de respuesta enviados por HTTP para ejectuar alguna accion

En visual studio code

En server.js

* Poner las siguiente instrucción ‘if’ en la funcion ‘app.post’ despues de ‘let’

app.post('/usuario', function(req, res) {

let body = req.body;

*if* (body.nombre === undefined) {

res.status(400).json({

ok: false,

mensaje: 'El nombre es necesario'

});

} *else* {

res.json({

persona: body

});

}

});

Esta instrucción checara si al recibirse del cliente una peticion ‘post’ , viene el nombre en el body de la peticion, esto se detecta si se recibio el codigo 400 desde HTTP; de no recibirse el nombre se enviara una respuesta indicandolo

En postman

* En el body quitar el ‘check’ en el recuadro del ‘nombre’
* Click a ‘send’

En el status aparece el codigo 400

En la ventana de body se despliegan los textos indicados en la instrucción ‘if’, en el objeto ‘res.status’

**CLASE 83 . Creando un archivo de configuracion global**

Objetivo: crear un archivo que nos permita cambiar del ambiente de desarrollo al de produccion (cambiando el puerto a escuchar) sin tener que modificar el codigo en los programas que levantan el servidor en cada uno de los ambientes

En visual studio code

* En folder ‘server’ crear folder ‘config’
* En folder ‘config’ crear archivo ‘config.js’

En archivo ‘config.js’

* Poner esta linea

process.env.PORT = process.env.PORT || 3000;

En archivo ‘server.js’

* Al inicio poner esta linea

require('./config/config');

Esta instrucción le indica al programa que lea y ejecute el archivo ‘config.js’ en el cual esta la instrucción de ‘process.env.PORT’ puesta para designar el puerto a escuchar

* Modificar estas lineas como se indica

app.listen(process.env.PORT, () => {

    console.log('Escuchando puerto: ', process.env.PORT);

En consola

* Teclear

Cd C:\Users\EG\_Lobo\Documents\Mis documentos\aa Major 1 EGB\EGB EDU\TAKEN\NODE\07-restserver

* Teclear: ‘nodemon server/server’ (para levantar servidor)
* Se despliegan varias lineas
* Se despliega el mensaje ‘Escuchando puerto: 3000’

En visual studio code

En archivo ‘config.js’

* Modificar esta linea como se indica

process.env.PORT = process.env.PORT || 3001;

En consola

* Se despliegan varias lineas
* Se despliega el mensaje ‘Escuchando puerto: 3001’

En visual studio code

En archivo ‘config.js’

* Modificar esta linea como se indica

process.env.PORT = process.env.PORT || 3000;

En consola

* Se despliegan varias lineas
* Se despliega el mensaje ‘Escuchando puerto: 3000’

FIN