1. Crear carpeta del proyecto
2. Ejecutar en consola el comando ***npm init –yes*** para crear el archivo package.json que es el archivo principal de todo el proyecto, el cual escribe toda la meta información del proyecto como también sus dependencias de los modulos a usar
3. Instalación de módulos a utilizar:

* **Express** (Framework de node js)
* **Mysql** (Gestor de base de datos)
* **Express-myconnection (**para llevar la conexión crud del proyecto, permitirá escribir el codigo de sql mas rapido**)**
* **Morgan (**servirá para ver lo que un usuario está pidiendo al ingresar a la servidor, si es una ruta especifica por ejemplo)
* **Ejs (**motor de plantilla)

con el comando ***npm install express mysql express-myconnection morgan ejs.***

1. Crear la carpeta src en la raiz del poyecto y dentro el archivo app.js
2. En el archivo app escribir el siguiente codigo para inicializar el archivo app

//requerir express, llamando el modulo express

const express=require('express');

const app=express();

//inicializando el archivo app

app.listen(3000/\*puerto\*/,()=>{

    console.log("Este es el servidor 3000");

});

1. Iniciar el servidor con el comando ***node src/app.js***
2. Una vez corriendo el servidor, se puede correr en el navegador de tu pc el servidor local con el puerto 3000. <http://localhost:3000/>
3. Al correrlo dara un error o un texto “Cannot Get”(Esto es normal)
4. Instalar en el proyecto el modulo “Nodemon” (Este servirá para que nuestro proyecto se ejecute en el servidor automáticamente, evitando el comando anterior:node para reiniciar el servidor ya que lo hará por defecto) con el comando ***npm install nodemon -D***
5. Si revisas el archivo pckage.json veras que nodemon no está en las dependencias, ya que esta es una dependencia de desarrollo la cual ira en otra área y es la que se agrega en la parte de debajo de las dependencias
6. Establecer el puerto del servidor en el archivo app.js con la siguiente línea de codigo

app.set('port', process.env.PORT || 3000);

1. Iniciar el servidor, ahora ya no sera con el commando node, sino con el commando nodemon src/app.js
2. Dara un error, pero eso es normal, para solucionar haremos de esta dependencia de desarrollador una global, para ello en el archivo package.json en scripts aremos lo siguiente.

"dev": "nodemon src/app.js"

1. Para corer nuestro Proyecto sera con el commando ***npm run dev***
2. De esta forma cada que hagamos un cambio en nuestro proyecto el servidor se reiniciara automáticamente.
3. Realizar las configuraciones de donde se encuentren las vistas de nuestro proyecto, para ello crearemos la carpeta Views en src
4. Ahora en el archivo app.js colocaremos la siguiente configuración:

app.set('view engine','ejs');//configuracion del motor de plantilla, express sabe que el motor a utilizar es ejs

1. Ahora indicaremos la dirección de nuestras vistas para ello requerir el modulo Path en nuestro proyecto

const path = require("path");

1. Luego colocar el codigo para obtener la ruta de las vistas

//Indicar donde se encuentran nuestras vistas

app.set('views', path.join(\_\_dirname, 'views'));

1. Ahora crearemos los apartados de middelwares(funciones que gestionan la informacion ntes de llamar las routes->peticiones) y routes(peticiones http)
2. En los middlewares crearemos las siguientes funciones, pero antes vamos a requerir Morgan para estas acciones:

const morgan=require('morgan');

1. Ahora usaremos Morgan para gestionar la acción (petición hecha por el cliente)

//middlewares(ejecutar funciones antes de llamar las peticiones, gestionar lo que va)

app.use(morgan('dev'));

1. Una vez creada esta configuración y para prueba se puede ir al navegador y hacer una petición GET a <http://localhost:3000/> en el navegador no pasa nada, pero en consola del ide puedes observer la informacion que te muestra, veras que muestra un codigo de estado 404 y el tiempo tardado en hacer la petición
2. Crear conexión con MySql con el siguiente codigo: constantes,

const mysql=require('mysql');

const myConnection=require('express-myconnection');

1. Antes de crear la conexión con mysql crearemos nuestra base de datos con la siguiente tabla, esto puede ser en el paquete de xampp:

CREATE TABLE `customer` (

  `id` int(6) NOT NULL,

  `name` varchar(50) NOT NULL,

  `address` varchar(100) NOT NULL,

  `phone` varchar(9) NOT NULL

)

1. Ahora crearemos la conexión a mysql en nuestro archivo app.js

//Conexion a MySql

app.use(myConnection(mysql,{

    host:'localhost',

    user:'root',

    password:'',

    port:3306,

    database:'crudnodejs'

},'single'/\*\*forma en la que nos conectaremos a mysql \*/));

1. Ahora crearemos las rutas de nuestros servicios, crear en src la carpeta routes y controllers
2. Dentro de la carpeta routes crearemos el archivo customer.js

const express=require('express');

const router=express.Router();

///Exportando la ruta

module.exports=router;

1. Crear la carpeta public dentro de src y agregar el siguiente codigo en el archivo app.js

//routes(peticiones http)

app.use('/',customerRoutes);

1. Agregar el archivo customerController.js en la carpeta controllers para su futura codificación
2. Ahora agregaremos el método GET en nuestra ruta de customer:

router.get('/',(req,res)=>{

    res.send('Hello development');

});

De esta manera tendríamos nuestro primer Hola Desarrollador

1. Pero automatizaremos mas este mensaje de la siguiente manera, en el archivo customerController agregar el codigo del método colocado en customer.

const controller={};

///hola desarrollador

controller.list=(req,res)=>{

    res.send('Hello development');

}

//importando

module.exports=controller;

1. Ahora importer el archive en customer para poder usar el metodo list de customerController

const customerController=require('../controllers/customerController');

1. Llamando el metodo list en customer:

router.get('/',customerController.list);

1. Metodo GET a la base de datos en el archivo customerController

//metodo get a la db

controller.list=(req,res)=>{

    req.getConnection((err,conn)=>{

        conn.query('SELECT\*FROM customer',(err,clientes)=>{

            if(err){

                res.json(err);

            }

            console.log(clientes);

            res.render('customers');

        })

    });

}

1. Si notas en consola al hacer una petición desde el navegador, se cargan los datos que tienes en tu db. Ahora seguiremos a crear el diseño de nuestras vistas, crearemos el siguiente codigo en el archivo customers.js de la carpeta Views
2. Crear la estructura básica de codigo html 5 y agregar el cdn de bootswatch accediendo al siguiente link: <https://bootswatch.com/> seleccionar el tema solar en el icono de despliegue abrir en una nueva ventana bootstrap.min.css

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>CRUD Node Js & MySql</title>

    <link rel="stylesheet" href="https://bootswatch.com/4/solar/bootstrap.min.css">

</head>

<body>

    <h1>Hello development</h1>

</body>

</html>

1. Ahora crearemos el diseño de nuestro menú de navegación

<div class="navbar navbar-expand-lg fixed-top navbar-dark bg-dark">

        <div class="container">

          <a href="/" class="navbar-brand">Home</a>

          <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarResponsive" aria-controls="navbarResponsive" aria-expanded="false" aria-label="Toggle navigation">

            <span class="navbar-toggler-icon"></span>

          </button>

          <div class="collapse navbar-collapse" id="navbarResponsive">

            <ul class="navbar-nav">

              <li class="nav-item">

                <a class="nav-link" href="../help/">Clientes</a>

              </li>

            </ul>

          </div>

        </div>

      </div>

1. A continuación separaremos nuestra codigo reutilizable a otro archivo, crearemos en la carpeta Views la carpeta partials y dentro de ella el archivo header.ejs y footer.ejs

Heder.ejs

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>CRUD Node Js & MySql</title>

    <link rel="stylesheet" href="https://bootswatch.com/4/solar/bootstrap.min.css">

</head>

<body>

     <nav class="navbar navbar-dark bg-dark">

      <a href="/" class="navbar-brand">HOME</a>

    </nav>

Footer.ejs

</body>

</html>

1. El archivo customers hasta este momento quedaría sin ningún codigo, para ello agregaremos los archivos del header y footer de la siguiente manera a customers:

<!--CODIGO DEL MOTOR DE VISTAS EJS-->

<!--Header-->

<% include partials/\_header %>

<!--Footer-->

<% include partials/\_footer %>

1. Creando el formulario para agregar registros a la db en customers: este codigo ira entre el header y el footer.

<div class="container">

    <div class="row mt-3">

        <div class="col-md-5">

           <div class="card">

               <div class="card-body">

                <form action="/insert" method="POST">

                    <div class="form-group">

                        <input type="text" name="name" placeholder="Insert your name" class="form-control">

                    </div>

                    <div class="form-group">

                        <input type="text" name="address" placeholder="Insert your address" class="form-control">

                    </div>

                    <div class="form-group">

                        <input type="text" name="phone" placeholder="Insert your phone number" class="form-control">

                    </div>

                    <button type="submit" class="btn btn-primary">Save Customer</button>

                </form>

               </div>

           </div>

        </div>

        <div class="col-md-7"></div>

    </div>

</div>

1. En el archivo app.js agregaremos un nuevo middleware:

//estamos descodificando la informacion que viene desde el formulario

app.use(express.urlencoded({extended:false}));

1. Ahora haremos funcionar el método insert que se manda a llamar en el formulario en el archivo de rutas customer en la carpeta routes y crear el respectivo método en el controlador:

Test:

//metodo post a la db

controller.insert=(req,res)=>{

   console.log(req.body);

   res.send('Hey');

};

Ahora llenar el formulario y enviar la información, veras que te aparece la palabra enviada en el navegador y en la consola del ide te aparecerá la información del objeto que se esta enviando.

1. Programando el método post(Insert) de nuestro controlador:

//metodo post a la db

controller.insert=(req,res)=>{

   //console.log(req.body);

    const data=req.body;

    req.getConnection((err,conn)=>{

        conn.query('INSERT INTO customer set ?',[data],(err,clientes)=>{

            console.log(clientes);

            res.redirect('/');

        })

    });

};

1. Crearemos ahora la vista para mostrar la lista de registros que tenemos en la db: en la columna de 7 crearla siguiente tabla:

<table class="table table-bordered table-hover">

                <thead>

                    <tr>

                        <th>ID</th>

                        <th>NAME</th>

                        <th>ADDRESS</th>

                        <th>PHONE NUMBER</th>

                        <th>OPTIONS</th>

                    </tr>

                </thead>

                <tbody>

                    <% if(data){%>

                        <% for(var i=0; i<data.length; i++){ %>

                            <tr>

                                <td><%= (i+1)%></td>

                                <td><%= data[i].name%></td>

                                <td><%= data[i].address%></td>

                                <td><%= data[i].phone%></td>

                                <td></td>

                            </tr>

                        <% }%>

                    <% }%>

                </tbody>

            </table>

1. Metodo delete a la db en el controlador y agregar la ruta en el archivo customer de routes:

**Ruta en customer de route:**

router.get('/delete/:id',customerController.delete);

**Agregar el boton delete en la table de datos en option:**

 <td>

                                    <a href=" /delete/<%= data[i].id%>

" class="btn btn-danger btn-sm">Delete</a>

                                </td>

**Metodo en el controlador:**

//metodo delete a la db

controller.delete=(req,res)=>{

    const {id}=req.params;//otra forma = const id=req.params.id;

    req.getConnection((err,conn)=>{

        conn.query('DELETE FROM customer WHERE id= ?',[id],(err,clientes)=>{

            //console.log(clientes);

            res.redirect('/');

        })

    });

 };

**Min1:14**