



NODE JS + EXPRESS + MYSQL

CRUD

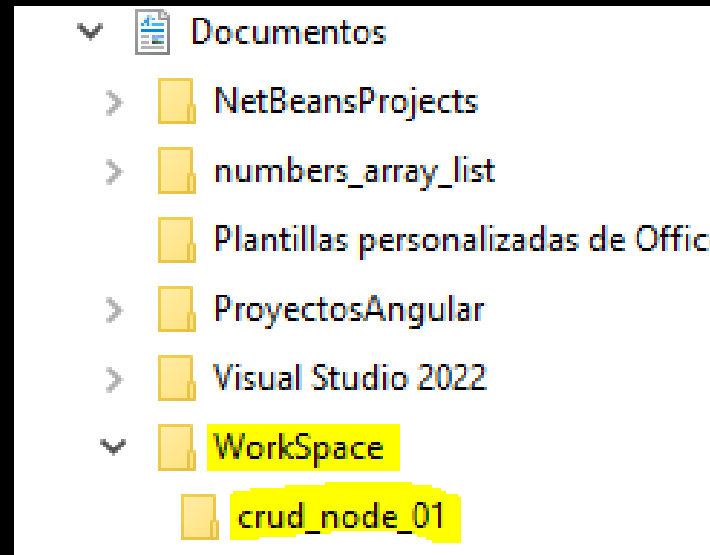
Nixon Duarte Acosta

PRIMERA PARTE

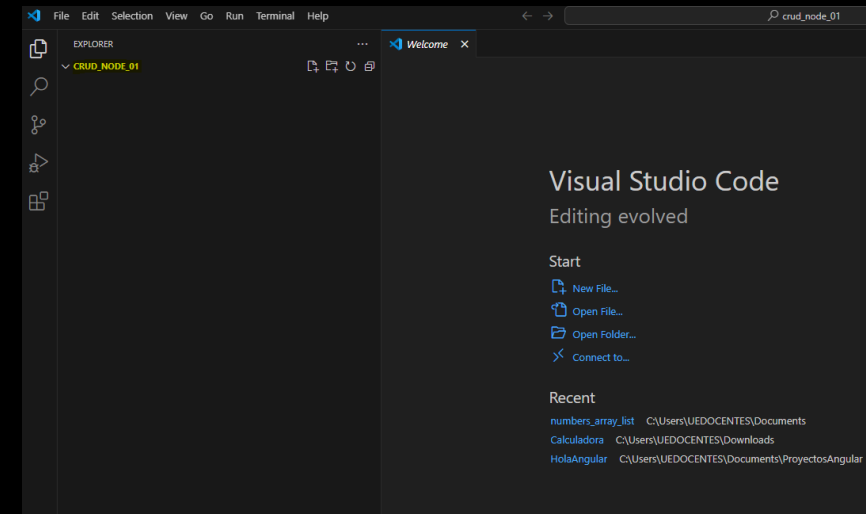
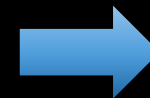


ESTRUCTURA DEL PROYECTO

1. Crear proyecto

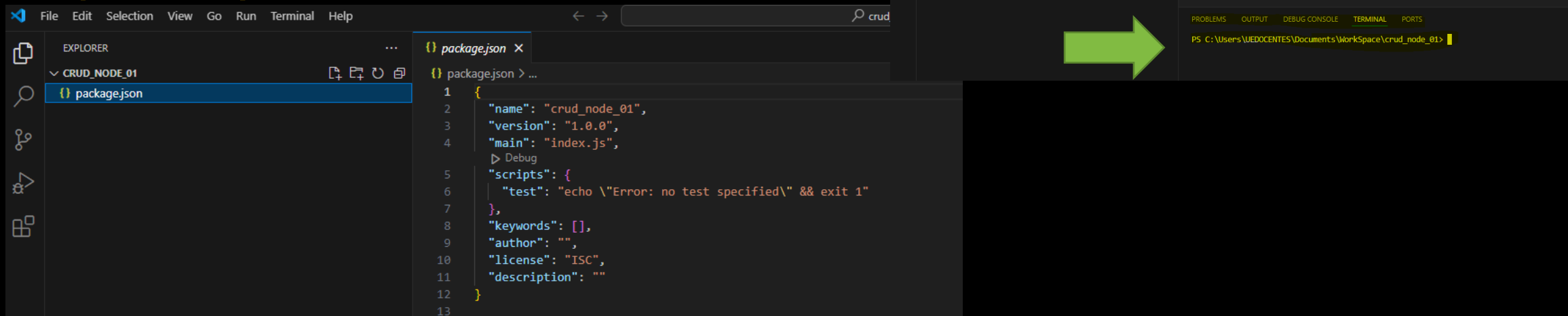


2. Abrir el folder en Visual Studio Code



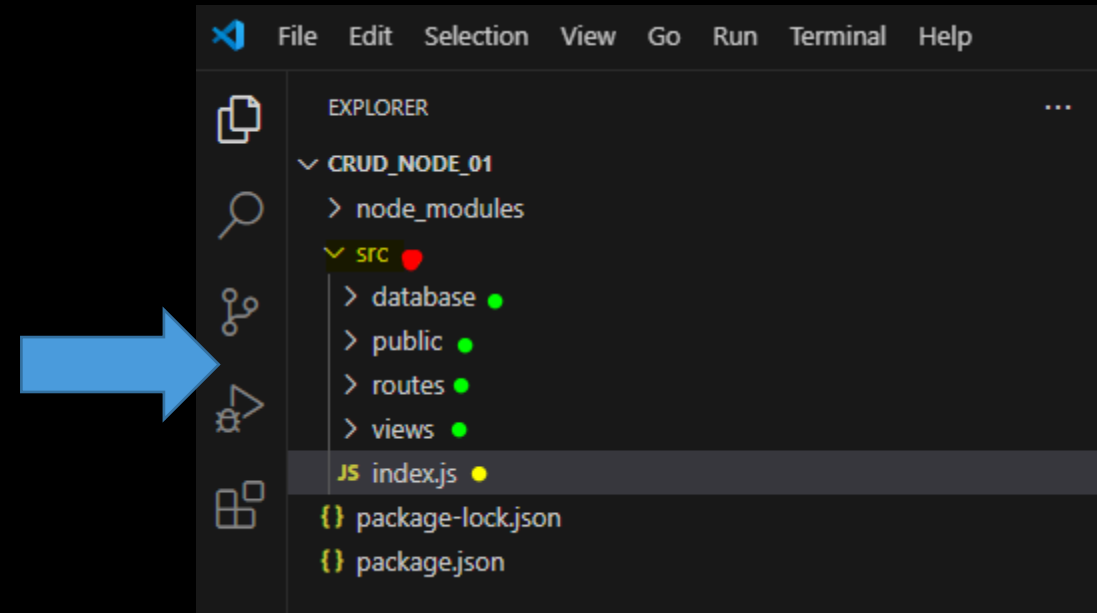
CONFIGURACIÓN DEL SERVIDOR

3. Abrir terminal en el IDE – VSC
 - Ctrl + shift + p
 - Buscar: *Create new terminal*
 - ...ó *abrir CMD* para ejecutar los comandos
4. Ejecutar comando para crear el `package.json`
 - **`npm init --y`**



CONFIGURACIÓN DEL SERVIDOR

5. Instalar los paquetes necesarios (librerías) que necesitamos para el funcionamiento de nuestro proyecto.
 - ***npm i express express-handlebars morgan mysql2***
 - Se genera la carpeta node_modules
6. Estructura del proyecto:



CONFIGURACIÓN DEL SERVIDOR

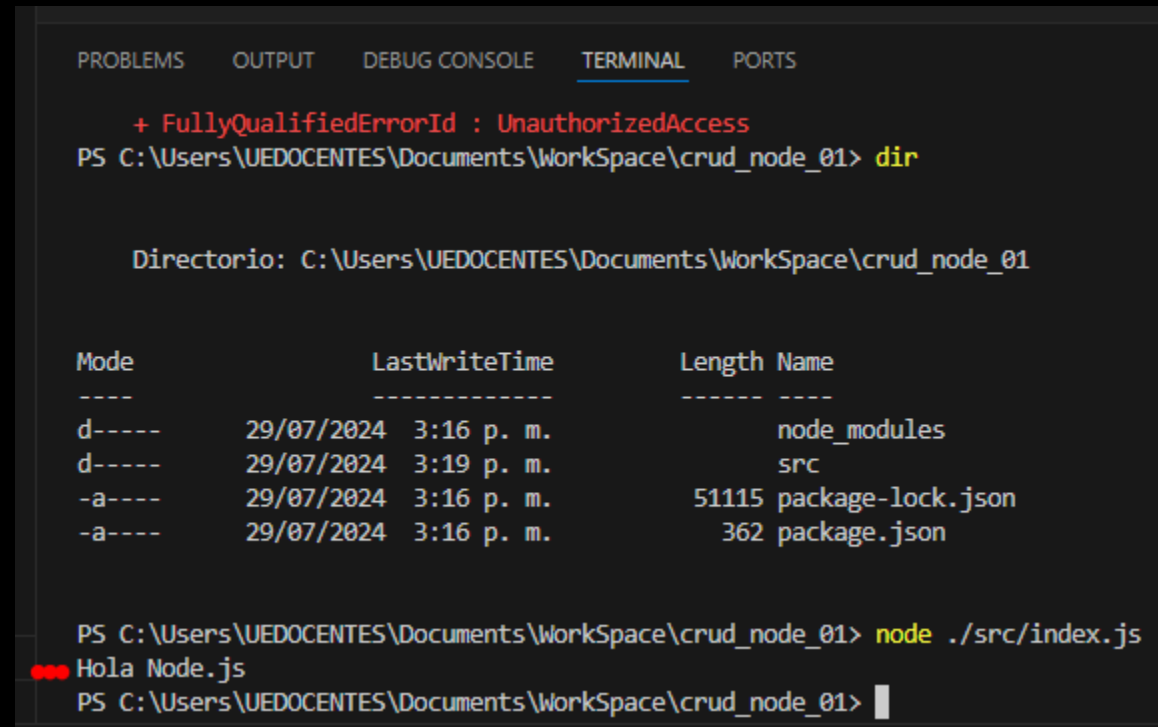
7. Crear *Hola Node.js*

- En *index.js*
 - Ingresar: **`console.log('Hola Node.js');`**
- En Terminal/CMD en el proyecto, ejecutar:

- **`node ./src/index.js`**

8. Instalar *nodemon* para facilitar la ejecución y no estar ejecutando el comando anterior:

- **`npm i nodemon -D`**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

+ FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01> dir

Directorio: C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01

Mode                LastWriteTime         Length Name
----                -
d-----          29/07/2024   3:16 p. m.      node_modules
d-----          29/07/2024   3:19 p. m.          src
-a----          29/07/2024   3:16 p. m.    51115 package-lock.json
-a----          29/07/2024   3:16 p. m.     362 package.json

PS C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01> node ./src/index.js
●●● Hola Node.js
PS C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01>
```

CONFIGURACIÓN DEL SERVIDOR

9. Modificar package.json

10. Ejecutamos:

- ***npm run dev***



```
▶ Debug  
"scripts": {  
  "dev": "nodemon ./src/index.js"  
},
```



```
C:\windows\system32\cmd.exe  
at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run_main:174:12)  
at node:internal/main/run_main_module:28:49 {  
  code: 'MODULE_NOT_FOUND',  
  requireStack: []  
}  
Node.js v20.13.0  
C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01>npm i nodemon -D  
added 28 packages, and audited 159 packages in 11s  
30 packages are looking for funding  
  run `npm fund` for details  
found 0 vulnerabilities  
C:\Users\UEDOCENTES\Documents\Workspace\crud_node_01>npm run dev  
> crud_node_01@1.0.0 dev  
> nodemon ./src/index.js  
[nodemon] 3.1.4  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting `node ./src/index.js`  
• Hola Node.js •  
[nodemon] clean exit - waiting for changes before restart
```


CONFIGURACIÓN DEL SERVIDOR

11. Modificamos el package.json

- Agregamos
 - **"type": "module",**

```
{ } package.json > ...
1  {
2    "name": "crud_node_01",
3    "version": "1.0.0",
4    "main": "index.js",
5    "type": "module",
6    "scripts": {
7      "dev": "nodemon ./src/index.js"
8    },
9  }
```

12. Estructuramos el index.js

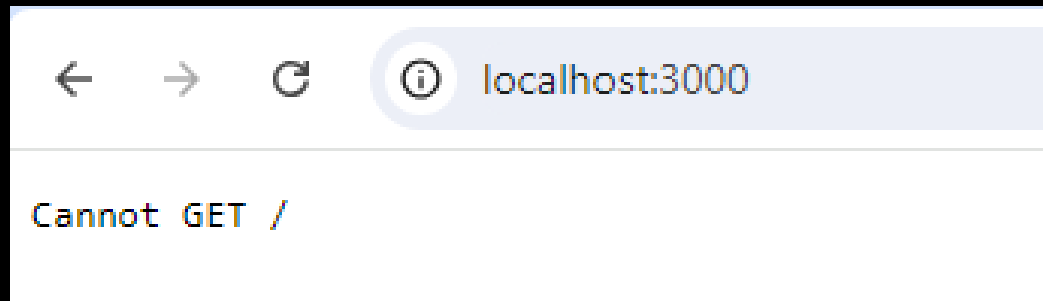
```
JS index.js x
src > JS index.js > app.listen() callback
1  import express from 'express'
2
3  //Initialization
4  const app = express();
5
6  //Setting
7  app.set('port', process.env.PORT || 3000);
8
9  //Middlewares
10
11 //Routes
12
13 //Public files
14
15 //Run server
16 app.listen(app.get('port'), ()=>
17   console.log('El server esta escuchando en el puerto', app.get('port')));
```


CONFIGURACIÓN DEL SERVIDOR

13. Ejecutamos o si esta en ejecución se puede ver por consola el mensaje:

- *El server esta escuchando en el puerto 3000*

14. Ahora podemos ir al navegador web e ingresar: *localhost:3000*

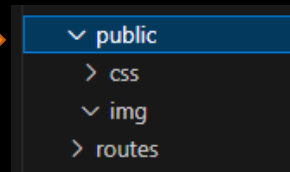
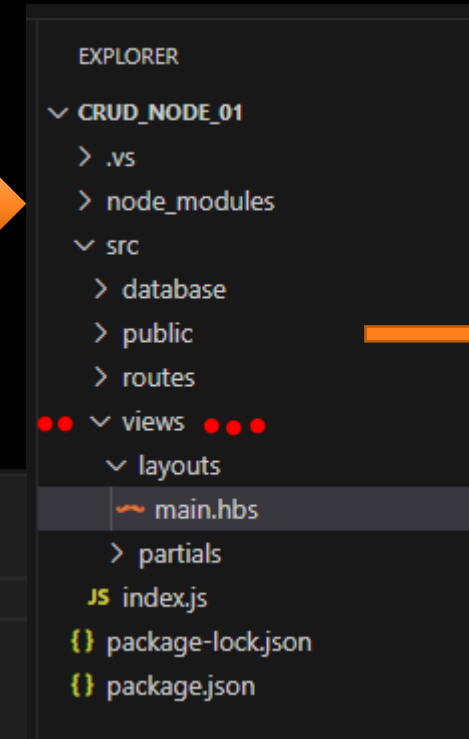


- *Esto significa que node.js ya se esta ejecutando, solo que no tenemos ningún routes.*

CONFIGURACIÓN DEL SERVIDOR

15. Agregamos en *views* las siguientes carpetas

16. Continuamos con la configuración del *main.hbs*




src > views > layouts > main.hbs > html


```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Document</title>
7 </head>
8 <body>
9   {{{body}}}
10 </body>
11 </html>
```

CONFIGURACIÓN DEL SERVIDOR

17. Continuamos con la configuración del *index.js*



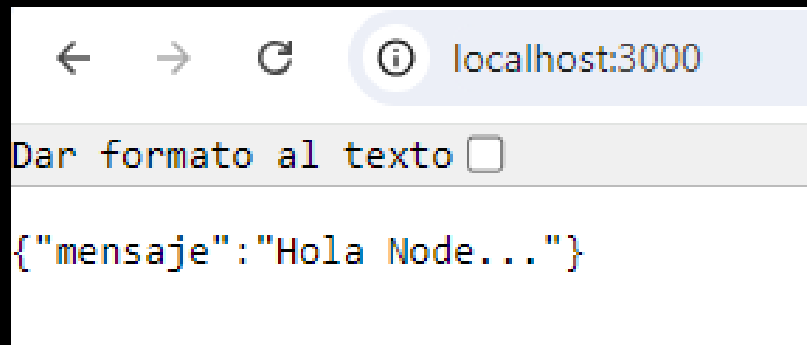
```
JS index.js X main.hbs
src > JS index.js > ...
1  import express from 'express'
2  import morgan from 'morgan';
3  import { engine } from 'express-handlebars';
4  import {join, dirname} from 'path'
5  import { fileURLToPath } from 'url';
6
7  //Initialization
8  const app = express();
9  const __dirname = dirname(fileURLToPath(import.meta.url));
10
11 //Setting
12 app.set('port', process.env.PORT || 3000);
13 app.set('views', join(__dirname, 'views'));
14 app.engine('.hbs', engine({
15   defaultLayout: 'main',
16   layoutsDir: join(app.get('views'), 'layouts'),
17   partialsDir: join(app.get('views'), 'partials'),
18   extname: '.hbs'
19 }));
20 app.set('view engine', '.hbs');
21
```



```
22
23 //Middlewares
24 app.use(morgan('dev'));
25 app.use(express.urlencoded({extended: false}));
26 app.use(express.json());
27
28 //Routes
29 app.get('/', (req, res) => {
30   res.json({"mensaje": "Hola Node..."})
31 });
32
33 //Public files
34 app.use(express.static(join(__dirname, 'public')));
35
36 //Run server
37 app.listen(app.get('port'), ()=>
38   console.log('El server esta escuchando en el puerto', app.get('port')));
```

CONFIGURACIÓN DEL SERVIDOR

- Al actualizar el navegador web, debe mostrar:



FIN PRIMERA PARTE



SEGUNDA PARTE

1. Continuamos con la configuración del *main.hbs*
 - Agregamos Bootstrap

```
JS index.js  main.hbs  index.hbs
src > views > layouts > main.hbs > html
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      {{!--Bootstrap--}}
7      <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
      Dwwykc2MPK8M2HN" crossorigin="anonymous">
8      <title>:: CRUD NODE.js ::</title>
9  </head>
10 <body>
11     {{{body}}}
12
13     {{!--Script Bootstrap--}}
14
15     <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
        crossorigin="anonymous"></script>
16 </body>
17 </html>
```

CONFIGURACIÓN VIEWS

2. Creamos *index.hbs*, en la carpeta *views*

```
src > views > index.hbs > main.mt-5
1  <main class="mt-5">
2    <div class="container">
3      <div class="card mx-auto" style="width:500px;">
4        <div class="card-header">
5          
6          <h3 class="text-uppercase text-center p-4">CRUD con node js y mysql</h3>
7        </div>
8        <div class="card-body text-center">
9          <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Repellendus maiores pariatur at placeat vel quam corporis quod odit itaque
          eveniet cupiditate.</p>
10         <a href="#" class="btn btn-primary">Crear una persona</a>
11       </div>
12     </div>
13   </div>
14 </main>
```


3. Modificar en *index.js*

```
28 //Routes
29 app.get('/', (req, res) => {
30   res.render('index')
31 });
```


CONFIGURACIÓN VIEWS


4. En el folder “*partials*” creamos el archivo *navigation.hbs*, con el siguiente contenido:

5. Agregar en *main.hbs*



```
</head>
<body>
••• {{>navigation}} •••
  {{{body}}}

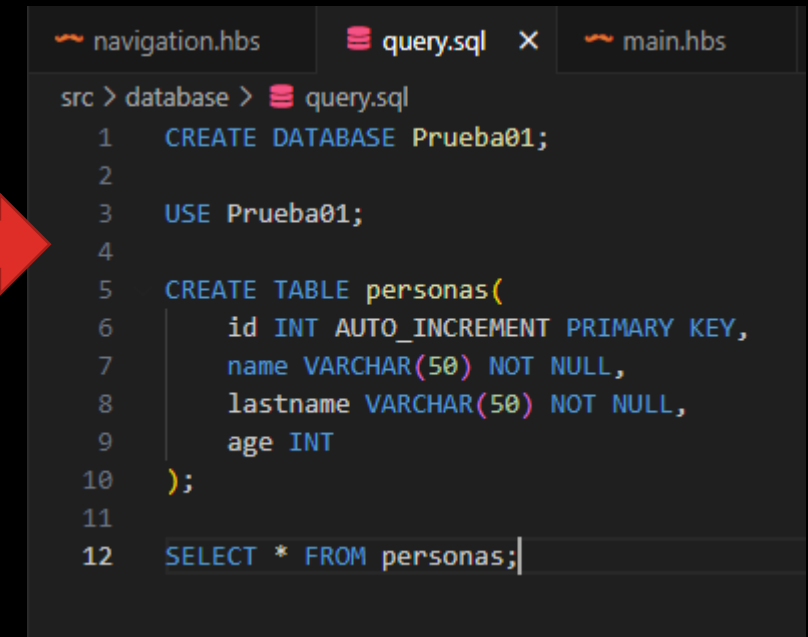
  {{!--Script Bootstrap--}}
```



```
navigation.hbs X main.hbs
src > views > partials > navigation.hbs > nav.navbar.navbar-expand-lg.navbar-dark.bg-dark > div.container-fluid > button.navbar-toggler
1 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
2   <div class="container-fluid">
3     <a class="navbar-brand" href="/">
4       <b>CRUD Node JS</b>
5     </a>
6     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"
7       aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
8       <span class="navbar-toggler-icon"></span>
9     </button>
10    <div class="collapse navbar-collapse" id="navbarNav">
11      <ul class="navbar-nav" style="margin-left: auto;">
12        <li class="nav-item">
13          <a class="nav-link" href="/">Inicio</a>
14        </li>
15        <li class="nav-item">
16          <a class="nav-link" href="/add">Crear</a>
17        </li>
18        <li class="nav-item">
19          <a class="nav-link" href="/list">Listar</a>
20        </li>
21      </ul>
22    </div>
23  </div>
24 </nav>
```

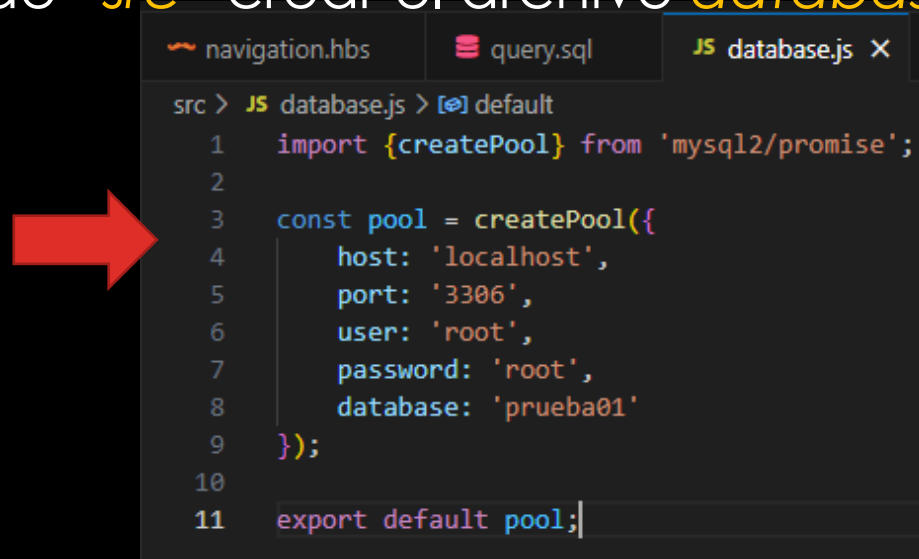
CREACIÓN DE LA BDS

6. Crear el folder “*database*” el archivo *query.sql*
7. Abrir MySQLWorkbench / phpMyAdmin
8. A nivel de “*src*” crear el archivo *database.js*



A screenshot of a code editor with three tabs: navigation.hbs, query.sql, and main.hbs. The query.sql tab is active, showing SQL code for creating a database and a table. A red arrow points from the text 'query.sql' in the list to this tab.

```
src > database > query.sql
1  CREATE DATABASE Prueba01;
2
3  USE Prueba01;
4
5  CREATE TABLE personas(
6      id INT AUTO_INCREMENT PRIMARY KEY,
7      name VARCHAR(50) NOT NULL,
8      lastname VARCHAR(50) NOT NULL,
9      age INT
10 );
11
12 SELECT * FROM personas;
```



A screenshot of a code editor with three tabs: navigation.hbs, query.sql, and JS database.js. The database.js tab is active, showing JavaScript code for creating a MySQL connection pool. A red arrow points from the text 'database.js' in the list to this tab.

```
src > JS database.js > default
1  import {createPool} from 'mysql2/promise';
2
3  const pool = createPool({
4      host: 'localhost',
5      port: '3306',
6      user: 'root',
7      password: 'root',
8      database: 'prueba01'
9  });
10
11 export default pool;
```

FIN SEGUNDA PARTE



TERCERA PARTE – LISTAR Y CREAR

1. En *views* creamos un folder *personas* y un archivo *list.hbs*

src > views > personas > list.hbs > div.container

```
1 <div class="container">
2   <div class="row mt-5">
3     <h3 class="text-center text-uppercase p-2">Lista de personas</h3>
4     {{#if personas}}
5       <table class="table text-center">
6         <thead>
7           <tr class="table-dark">
8             <th scope="col" class="col-4">Nombre</th>
9             <th scope="col">Apellido</th>
10            <th scope="col">Edad</th>
11            <th scope="col">Acciones</th>
12          </tr>
13        </thead>
14        <tbody>
15          {{#each personas}}
16            <tr>
17              <td>{{name}}</td>
18              <td>{{lastname}}</td>
19              <td>{{age}}</td>
20              <td>
21                <div class="btn-group">
22                  <a href="/edit/{{id}}" class="btn btn-warning p-2">
23                    <i class="fa-solid fa-pencil"></i></a>
24                  <a href="/delete/{{id}}" class="btn btn-danger p-2">
25                    <i class="fa-solid fa-trash"></i></a>
26                </div>
```

```
27              </td>
28            </tr>
29          {{/each}}
30        </tbody>
31      </table>
32      <div class="card text-center mx-auto" style="width: 350px;">
33        <div class="card-header">
34          <h3>Debes crear una persona</h3>
35        </div>
36        <div class="card-body">
37          <p>Presiona este boton para que puedas registrar a una persona en la aplicacion</p>
38          <a href="#" class="btn btn-primary">Ir a crear</a>
39        </div>
40      </div>
41    </div>
42  </div>
43  {{/if}}
44  </div>
45  </div>
```



LISTAR Y CREAR

2. En *routes* creamos el archivo *personas.routes.js*

```
src > routes > JS personas.routes.js > [x] default
```

```
1  import {Router} from 'express'
2  import pool from '../database.js'
3
4  const router = Router();
5
6  router.get('/add', (req,res)=>{
7    res.render('personas/add');
8  });
9
10 router.post('/add', async(req, res)=>{
11   try{
12     const {name, lastname, age} = req.body;
13     const newPersona = {
14       name, lastname, age
15     }
16     await pool.query('INSERT INTO personas SET ?', [newPersona]);
17     res.redirect('/list');
18   }
19   catch(err){
20     res.status(500).json({message:err.message});
21   }
22 });
23
```

```
24 router.get('/list', async(req, res)=>{
25   try{
26     const [result] = await pool.query('SELECT * FROM personas');
27     res.render('personas/list', {personas: result});
28   }
29   catch(err){
30     res.status(500).json({message:err.message});
31   }
32 });
33
34 router.get('/edit/:id', async(req, res)=>{
35   try{
36     const {id} = req.params;
37     const [persona] = await pool.query('SELECT * FROM personas WHERE id = ?', [id]);
38     const personaEdit = persona[0];
39     res.render('personas/edit', {persona: personaEdit});
40   }
41   catch(err){
42     res.status(500).json({message:err.message});
43   }
44 })
45
```

LISTAR Y CREAR

2. En *routes* creamos el archivo *personas.routes.js*

```
46 router.post('/edit/:id', async(req, res)=>{
47   try{
48     const {name, lastname, age} = req.body;
49     const {id} = req.params;
50     const editPersona = {name, lastname, age};
51     await pool.query('UPDATE personas SET ? WHERE id = ?', [editPersona, id]);
52     res.redirect('/list');
53   }
54   catch(err){
55     res.status(500).json({message:err.message});
56   }
57 })
58
59 router.get('/delete/:id', async(req, res)=>{
60   try{
61     const {id} = req.params;
62     await pool.query('DELETE FROM personas WHERE id = ?', [id]);
63     res.redirect('/list');
64   }
65   catch(err){
66     res.status(500).json({message:err.message});
67   }
68 });
69 export default router;
```


LISTAR Y CREAR

3. En *personas* creamos el archivo *add.hbs*

```
src > views > personas > add.hbs > div.container.p-4
1 <div class="container p-4">
2   <div class="row">
3     <div class="col-md-4 mx-auto">
4       <div class="card text-center">
5         <div class="card-header">
6           <h3 class="text-uppercase">CREAR NUEVA PERSONA</h3>
7         </div>
8         <div class="card-body">
9           <form action="/add" method="post">
10            <div class="input-group mt-2">
11              <label for="name" class="input-group-text">Nombre</label>
12              <input class="form-control" type="text" name="name" id="name" placeholder="Ejemplo: Luis Angel" autofocus required>
13            </div>
14            <div class="input-group mt-2">
15              <label for="lastname" class="input-group-text">Apellido</label>
16              <input class="form-control" type="text" name="lastname" id="lastname" placeholder="Ejemplo: Fernandez" required>
17            </div>
18            <div class="input-group mt-2">
19              <label for="age" class="input-group-text">Edad</label>
20              <input class="form-control" type="number" name="age" id="age" placeholder="0" required>
21            </div>
22            <div class="form-group mt-4 d-grid gap-2">
23              <button class="btn btn-success"> Crear</button>
24            </div>
25          </form>
26        </div>
27      </div>
28    </div>
29  </div>
30 </div>
```


LISTAR Y CREAR

3. En *personas* creamos el archivo *edit.hbs*

```
src > views > personas > edit.hbs > div.container.p-4
1 <div class="conatiner p-4">
2   <div class="row">
3     <div class="col-md-4 mx-auto">
4       <div class="card text-center">
5         <div class="card-header">
6           <h3 class="text-uppercase">EDITANDO UNA PERSONA</h3>
7         </div>
8         <div class="card-body">
9           <form action="/edit/{{persona.id}}" method="post">
10            <div class="input-group mt-2">
11              <label for="name" class="input-group-text">Nombre</label>
12              <input class="form-control" type="text" name="name" id="name" value="{{persona.name}}" placeholder="Ejemplo: Luis Angel" autofocus required>
13            </div>
14            <div class="input-group mt-2">
15              <label for="lastname" class="input-group-text">Apellido</label>
16              <input class="form-control" type="text" name="lastname" id="lastname" placeholder="Ejemplo: Fernandez" value="{{persona.lastname}}" required>
17            </div>
18            <div class="input-group mt-2">
19              <label for="age" class="input-group-text">Edad</label>
20              <input class="form-control" type="number" name="age" id="age" value="{{persona.age}}" placeholder="0" required>
21            </div>
22            <div class="form-group mt-4 d-grid gap-2">
23              <button class="btn btn-success"> Actualizar</button>
24            </div>
25          </form>
26        </div>
27      </div>
28    </div>
29  </div>
30 </div>
```

LISTAR Y CREAR

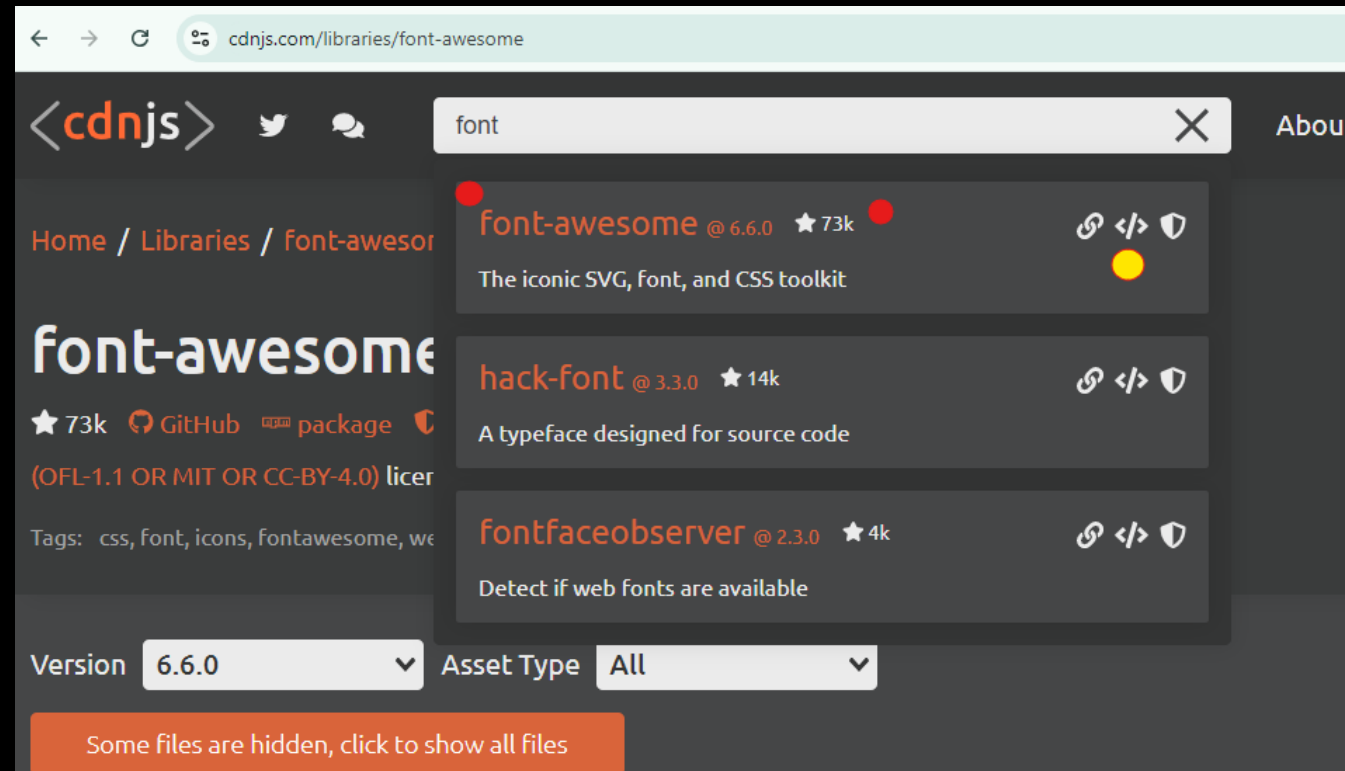
4. Agregar al *index.js* lo siguiente:

```
5   import { fileURLToPath } from 'url';  
6   import personasRoutes from '../routes/personas.routes.js'
```

```
29  //Routes  
30  app.get('/', (req, res) => {  
31    |    res.render('index')  
32  });  
33  
34  app.use(personasRoutes);  
35  
36  //Public files
```

LISTAR Y CREAR

4. Para utilizar iconos, se debe buscar en: <https://cdnjs.com/libraries/font-awesome>, y se busca “Font-awesome” y se copia el link, para posteriormente agregar al *main.hbs*, en la sección del *head*



The screenshot shows the cdnjs.com website with the search results for 'font'. The page has a dark theme. At the top, there's a search bar with 'font' entered. Below the search bar, there are three results listed:

- font-awesome** @ 6.6.0 ★ 73k. Description: The iconic SVG, font, and CSS toolkit.
- hack-font** @ 3.3.0 ★ 14k. Description: A typeface designed for source code.
- fontfaceobserver** @ 2.3.0 ★ 4k. Description: Detect if web fonts are available.

At the bottom of the page, there are filters for 'Version' (6.6.0) and 'Asset Type' (All). A button at the bottom says 'Some files are hidden, click to show all files'.

LISTAR Y CREAR

5. Para buscar imágenes de iconos, se puede hacer en:


- <https://fontawesome.com/icons>
- Se puede buscar los iconos como los utilizados en *list.hbs*
- El *pencil* para edición y el icono de *trash* para eliminar.



LISTAR Y CREAR

6. Al ejecutar se debe ver algo así:

CRUD Node JS

 Inicio Crear Listar



CRUD CON NODE JS Y MYSQL

Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Repellendus maiores pariatur at placeat vel quam corporis quod
odit itaque illum, non neque, perferendis eveniet cupiditate.

Crear una persona

LISTAR Y CREAR

6. Al ejecutar se debe ver algo así:



CRUD Node JS

[Inicio](#) [Crear](#) [Listar](#)

CREAR NUEVA PERSONA

Nombre	<input type="text" value="Ejemplo: Luis Angel"/>
Apellido	<input type="text" value="Ejemplo: Fernandez"/>
Edad	<input type="text" value="0"/>

Crear

LISTAR Y CREAR

6. Al ejecutar se debe ver algo así:

CRUD Node JS

Inicio Crear Listar

LISTA DE PERSONAS

Nombre	Apellido	Edad	Acciones
Jose Jose	Diaz	25	 

REGISTRO DE PERSONAS

FIN TERCERA PARTE

