

Ejercicio Supabase

Caren Juliana Arevalo Vargas

ID: 838908

Corporación Universitaria Minuto de Dios

William Alexander Matallana Porras

Bases de Datos Masivas

Abril de 2025

TABLA DE CONTENIDO

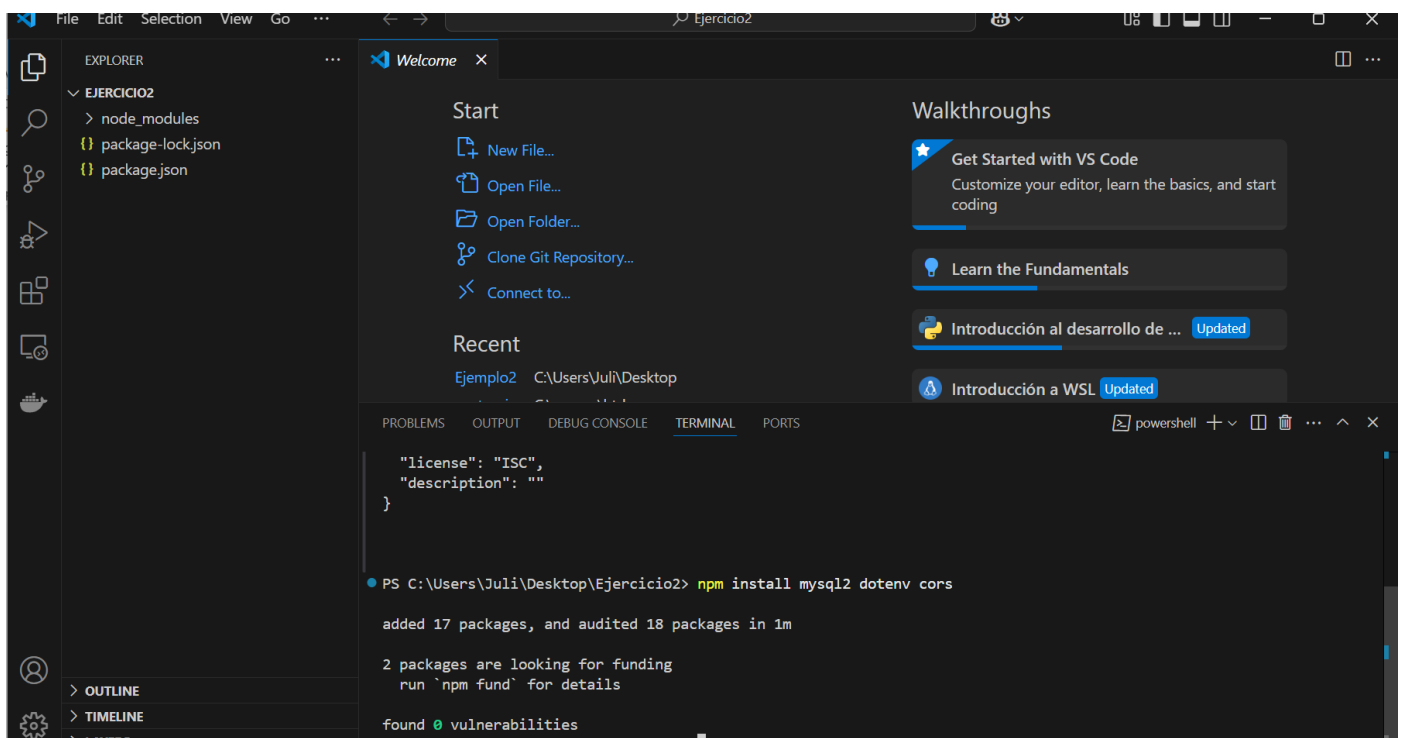
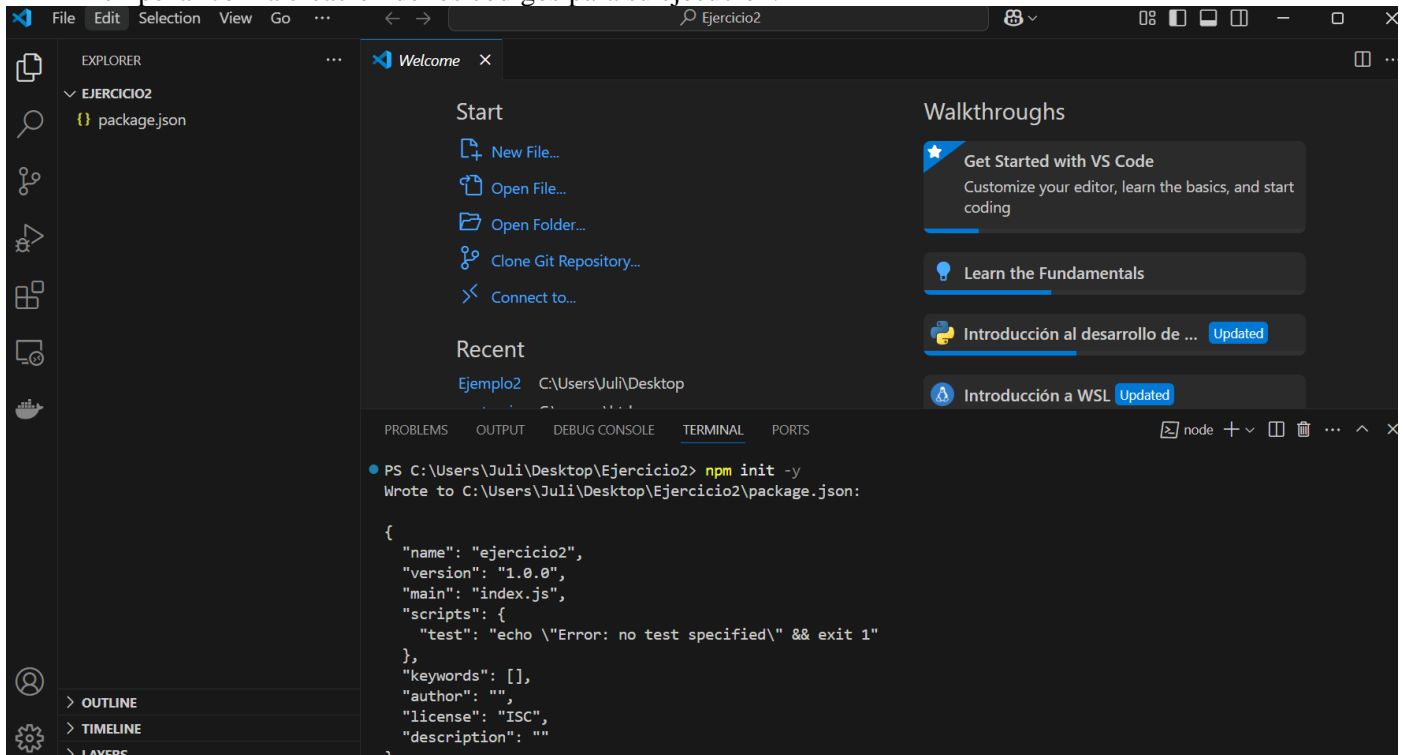
Introducción

1. Creación del nuevo contenedor Docker: Vamos a crear el nuevo contenedor Docker

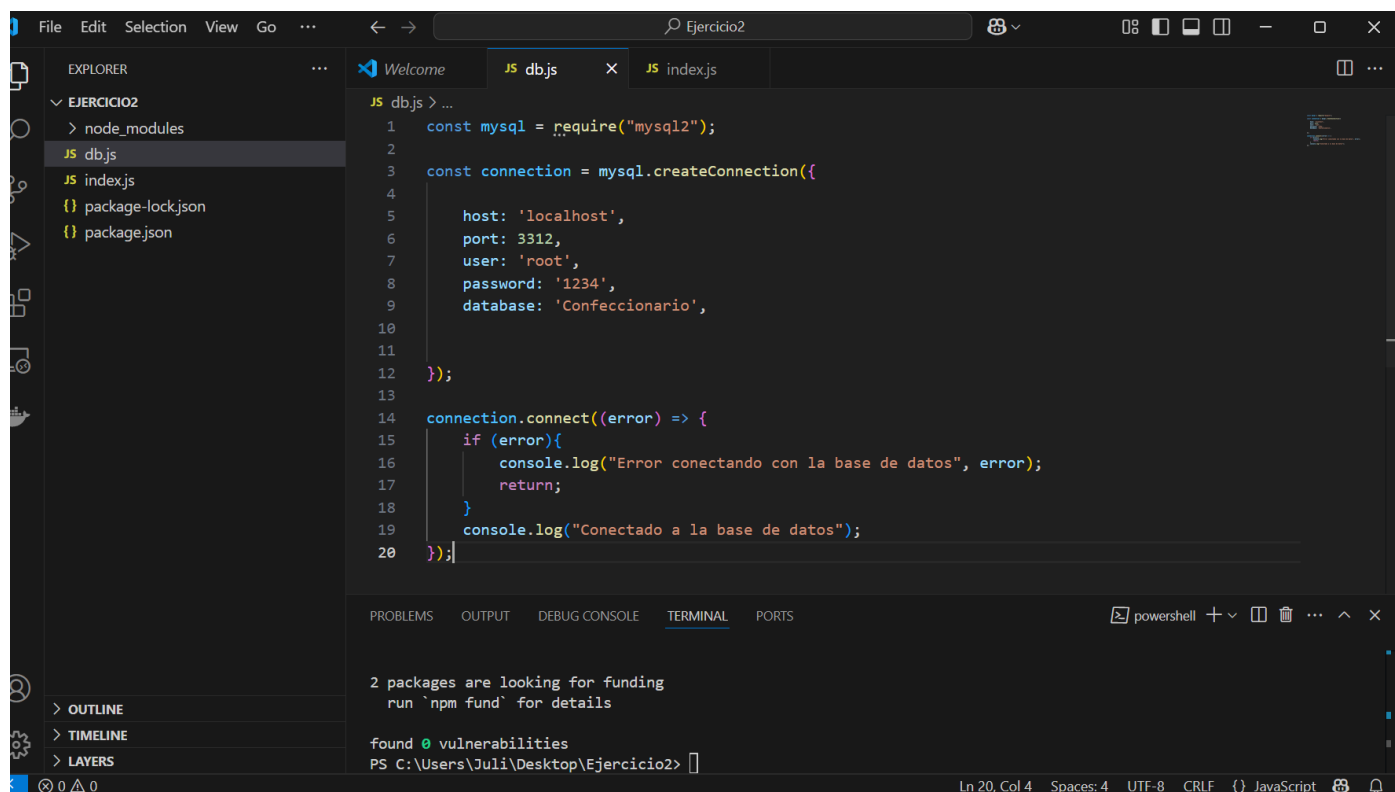
```
C:\Users\Juli>docker run -d -p 3312:3306 --name Ejercicio2 -e MYSQL_ROOT_PASSWORD=1234 mysql:latest
47a41198528f409e922e1dae22b776b902a7631a667fc3ca5afc9c82e44fd834
```

```
C:\Users\Juli>
```

2. Creación de la carpeta para los códigos: Crearemos la carpeta que se abrirá con visual studio code para empezar con la creación de los códigos para su ejecución.



Empezaremos a crear los index y demás para empezar a realizar la conexión

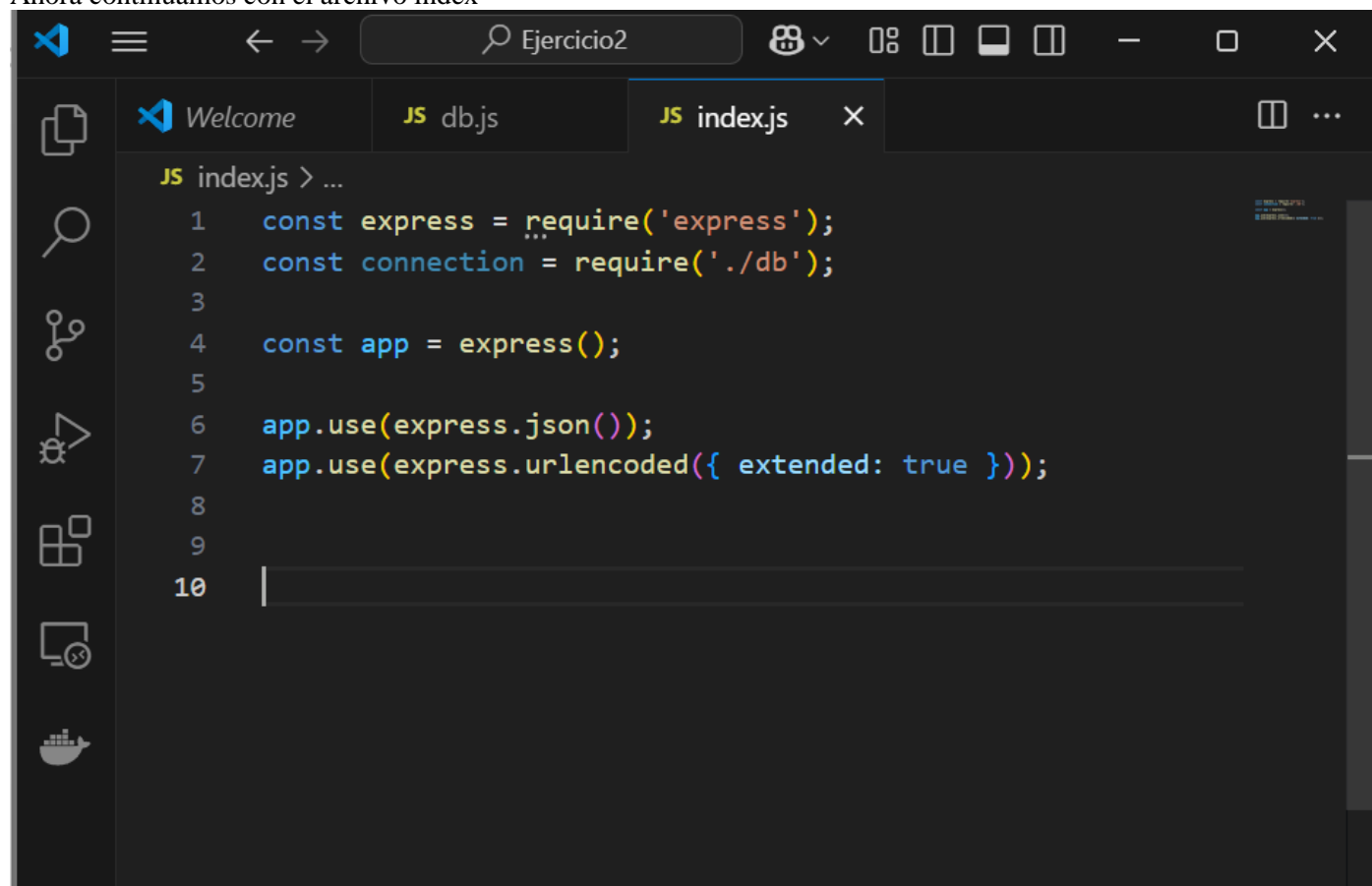


The screenshot shows the Visual Studio Code editor with a project named 'Ejercicio2'. The Explorer sidebar on the left shows the file structure: 'node_modules', 'db.js', 'index.js', 'package-lock.json', and 'package.json'. The main editor window displays the content of 'db.js', which contains the following JavaScript code:

```
1 const mysql = require("mysql2");
2
3 const connection = mysql.createConnection({
4
5     host: 'localhost',
6     port: 3312,
7     user: 'root',
8     password: '1234',
9     database: 'Confecionario',
10
11 });
12
13
14 connection.connect((error) => {
15     if (error){
16         console.log("Error conectando con la base de datos", error);
17         return;
18     }
19     console.log("Conectado a la base de datos");
20 });
```

At the bottom of the editor, the TERMINAL panel shows the output of running 'npm fund', indicating that two packages are looking for funding and that no vulnerabilities were found.

Ahora continuamos con el archivo index

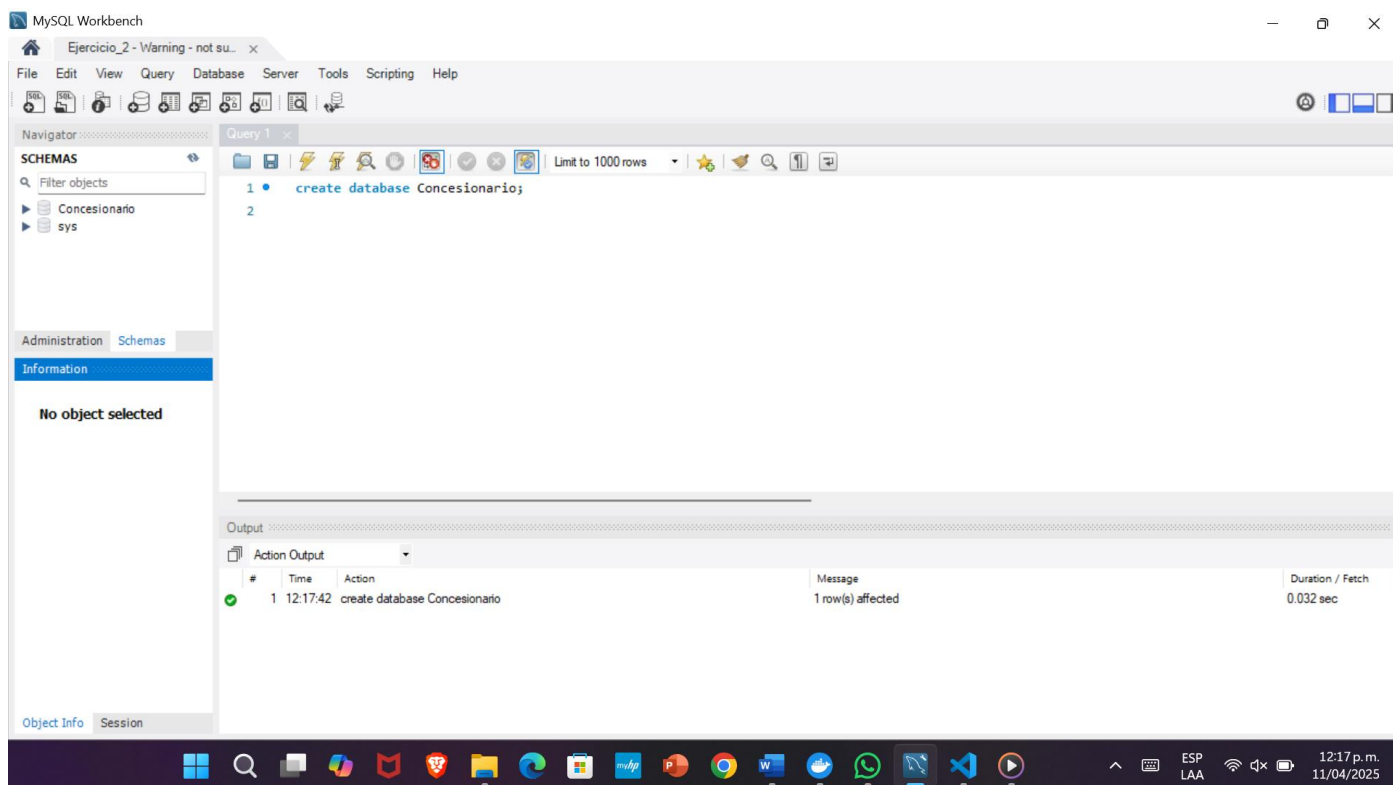


The screenshot shows the Visual Studio Code editor with the 'index.js' file open. The code in the file is as follows:

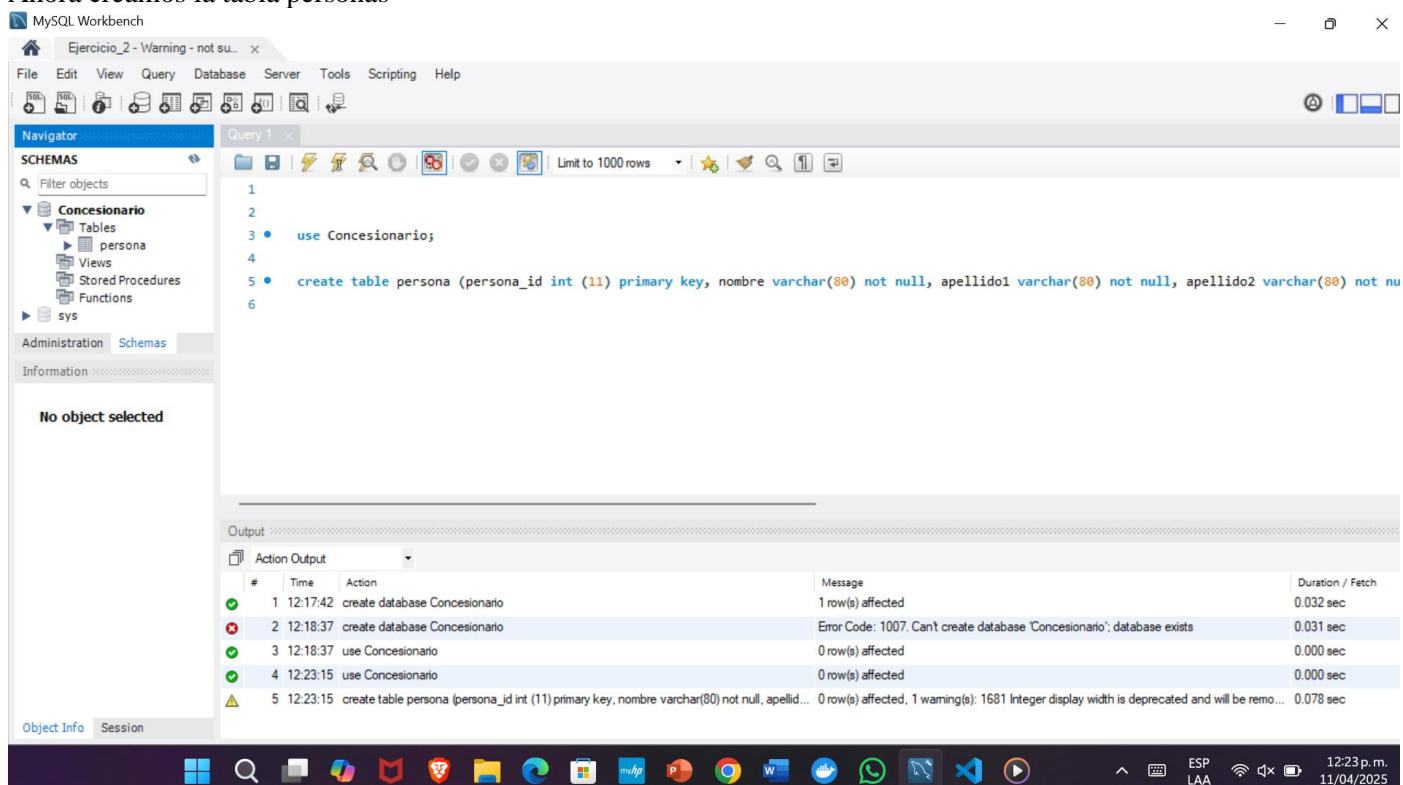
```
1 const express = require('express');
2 const connection = require('./db');
3
4 const app = express();
5
6 app.use(express.json());
7 app.use(express.urlencoded({ extended: true }));
8
9
10 |
```

The cursor is positioned at the end of line 10, ready for further code entry.

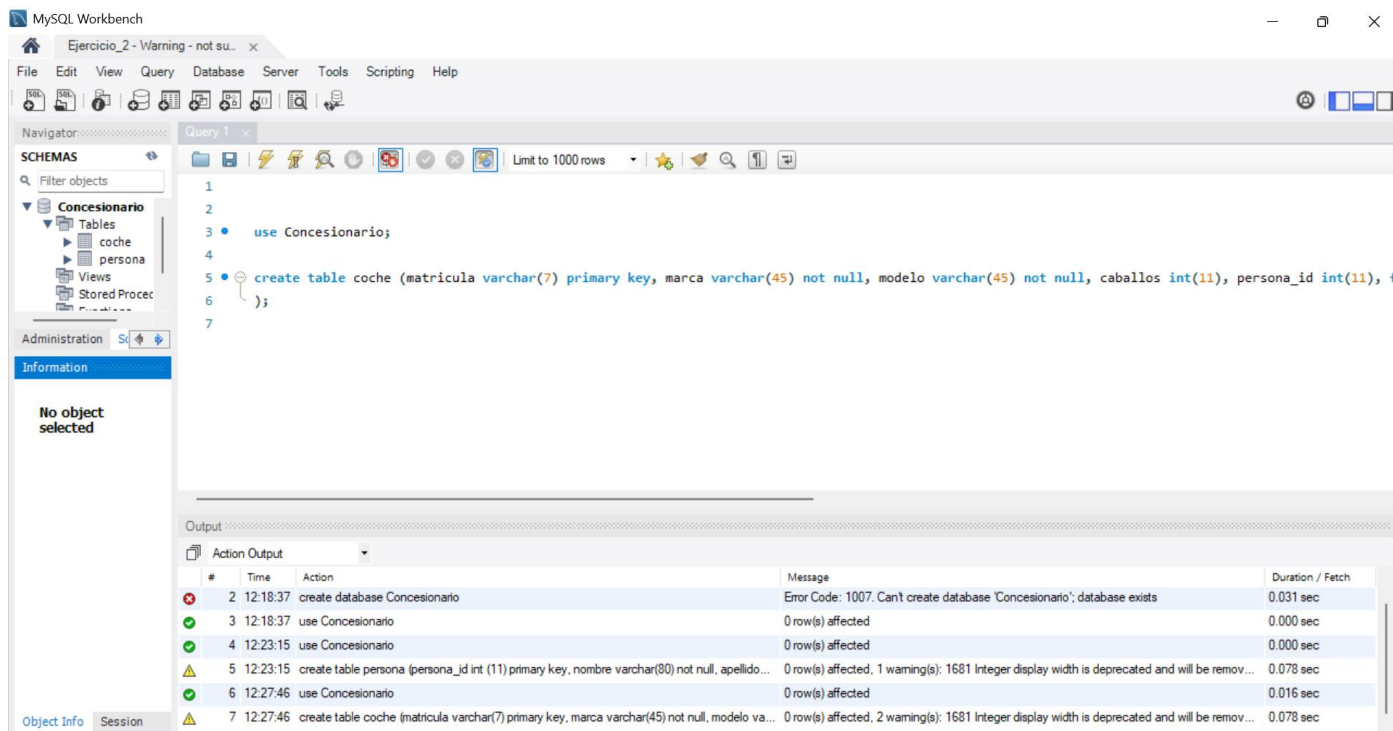
Antes de continuar con la creación de nuestro otro archivo vamos a crear la base de datos en mysql workbench



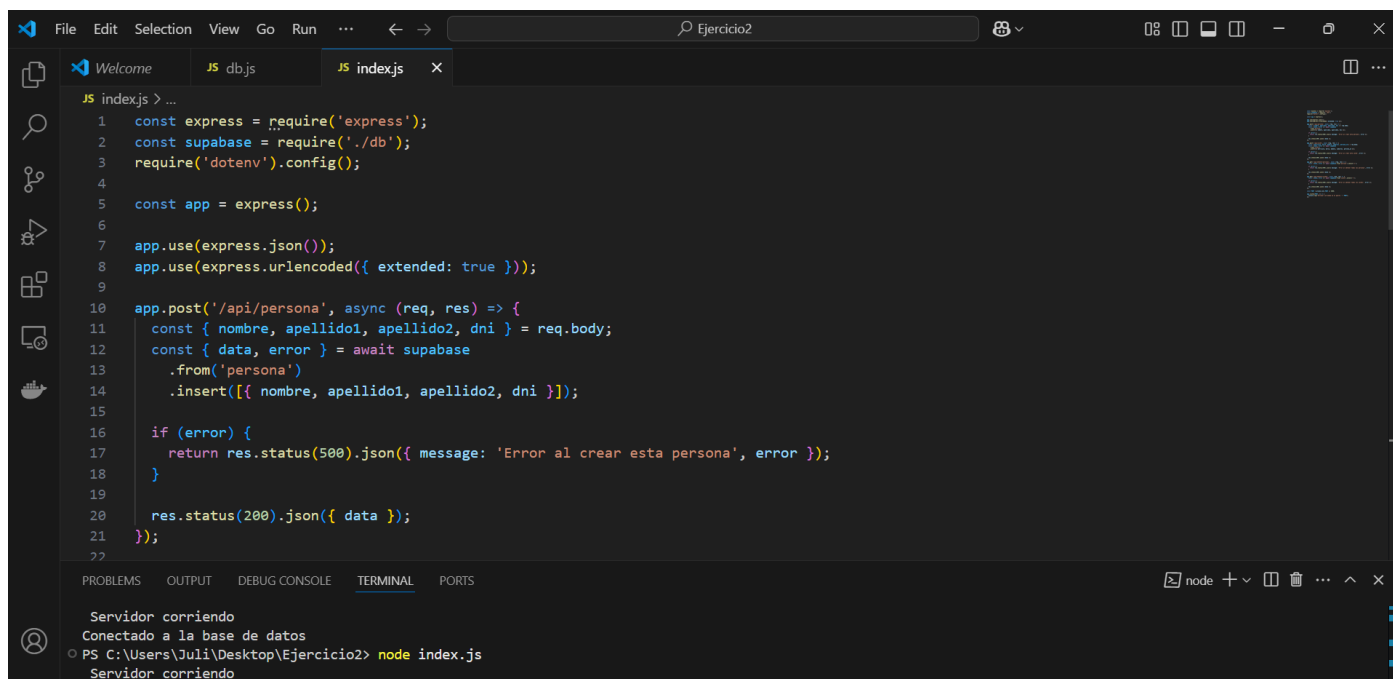
Ahora creamos la tabla personas



Y continuamos con la creación de la tabla coche



Ahora podemos continuar con nuestro archivo index y la primera api que creamos es la api de persona



Y seguimos con las otras crud

```

22
23 app.post('/api/coche', async (req, res) => {
24   const { matricula, marca, modelo, caballos, persona_id } = req.body;
25   const { data, error } = await supabase
26     .from('coche')
27     .insert([[{ matricula, marca, modelo, caballos, persona_id }]]);
28
29   if (error) {
30     return res.status(500).json({ message: 'Error al crear este coche', error });
31   }
32
33   res.status(200).json({ data });
34 });
35
36 app.get('/api/obtener/personas', async (req, res) => {
37   const { data, error } = await supabase.from('persona').select('*');
38
39   if (error) {
40     return res.status(500).json({ message: 'Error al obtener todas las personas', error });
41   }
42
43   res.status(200).json({ data });
44 });
45
46 app.get('/api/obtener/coches', async (req, res) => {
47   const { data, error } = await supabase.from('coche').select('*');
48
49   if (error) {
50     return res.status(500).json({ message: 'Error al obtener todos los coches', error });
51   }
52
53   res.status(200).json({ data });
54 });
55
56 const PORT = process.env.PORT || 3000;
57
58 app.listen(PORT, () => {
59   console.log('Servidor corriendo en el puerto ' + PORT);
60 });

```