

Manipular base de datos con Workbench y POSTMAN

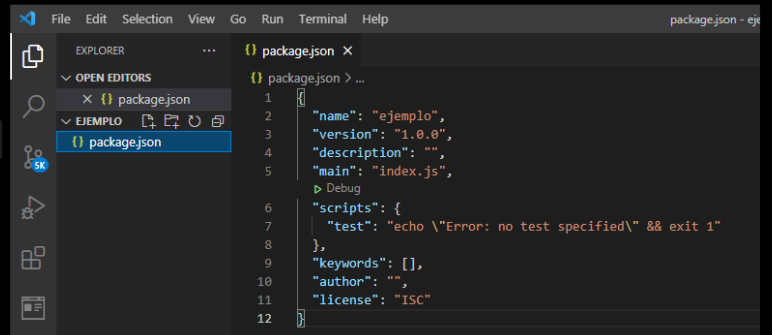
Objetivo: integrar las herramientas obtenidas de las materias Base de Datos y Programación WEB2.

Para lograr el objetivo comenzamos a crear un entorno de trabajo ☺

Escribir en terminal:

```
npm init -y
```

(se genera automaticamente el package.json)



```
1 {
2   "name": "ejemplo",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 }
```

```
{ } package.json > ...
```

```
1 {
2   "name": "ejemplo",
3   "version": "1.0.0",
4   "description": "The main field is a module ID that is the primary entry point to your program.",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC"
12 }
```

Modificamos main.js por app.js

```
"main": "app.js",
```

EXPLORER

OPEN EDITORS

{ } package.json

X JS app.js

EJEMPLO

JS app.js

{ } package.json

{ } package.json

JS app.js

JS app.js

1

Creamos un archivo con nombre: app.js

Para interactuar con nuestra de base de datos sql necesitamos instalar una dependencia: mysql:

npmjs.com/package/mysql

2.18.1 • Public • Published a year ago

Readme Explore BETA

La documentación la podemos encontrar en:
<https://www.npmjs.com/package/mysql>

mysql

npm v2.18.1 downloads 3.27M/month node >= 0.6 travis passing windows success coverage 98%

Install

> npm i mysql
Copy Command to Clipboard

Weekly Downloads

Table of Contents

Para instalar la dependencia: `npm i mysql`

```
{
  "name": "ejemplo",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "mysql": "^2.18.1"
  }
}
```

```
{
  "dependencies": {
    "mysql": "^2.18.1"
  }
}
```

Luego instalamos como devDependencia: express , body-parser y nodemon (para no tener que estar reiniciando continuamente) .

```
npm i -D express nodemon body-parser
```

```

JS app.js {} package.json X
{} package.json > {} devDependencies > [ ] nodemon
1  {
2    "name": "ejemplo",
3    "version": "1.0.0",
4    "description": "",
5    "main": "app.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "dependencies": {
13     "mysql": "^2.18.1"
14   },
15   "devDependencies": {
16     "body-parser": "^1.19.0",
17     "express": "^4.17.1",
18     "nodemon": "^2.0.7"
19   }
20 }
21

```

```

"devDependencies": {
  "body-parser": "^1.19.0",
  "express": "^4.17.1",
  "nodemon": "^2.0.7"
}

```

Configurar nodemon en el package.json:

```

"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node app.js",
  "dev": "nodemon --watch app.js"
},

```

Iniciar nodemon desde la terminal: `npm run dev`

Si modificamos el archivo `app.js`, no tenemos que reiniciar el servidor dado que Nodemon está observando los cambios mediante su parámetro `--watch`.
Se actualiza solo sin tener que reiniciar el servidor, para cortarlo presionar `CTRL + C`

Programando app.js

```
// requerir o importar express
const express = require ('express');
// requerir o importar mysql
const mysql = require ('mysql');
// requerir o importar body-parser
const bodyParser = require ('body-parser');

// generar una constante para nuestro puerto
const PORT = 4000;

// crear constante llamada app y generamos una instancia de express
const app = express ();

// utilizamos app.use y le pasamos bodyparser
app.use (bodyParser.json ());

// mysql en base a la documentacion:
//https://www.npmjs.com/package/mysql

const connection = mysql.createConnection ({
  host: 'localhost',
  user: 'root',
  password: 'root',
  // En el password va tu contraseña que usas en el workbench
  database: 'mibase_sql',
});

// chequeo de conexión
connection.connect(error=>{
  if(error) throw error;
  console.info("la base de datos esta funcionando")
});

// escuchar el puerto (alt96 = `)
app.listen(PORT,()=>console.info(`El servidor esta escuchando en el puerto:${PORT}`));
```

si ejecutamos npm run dev en la terminal se puede observar que va iniciar el puerto, pero genera un error porque no tenemos una base de datos:

```
database: 'mibase_sql',
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo> npm run dev

> ejemplo@1.0.0 dev
> nodemon --watch app.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): app.js
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
El servidor esta escuchando en el puerto:4000
```

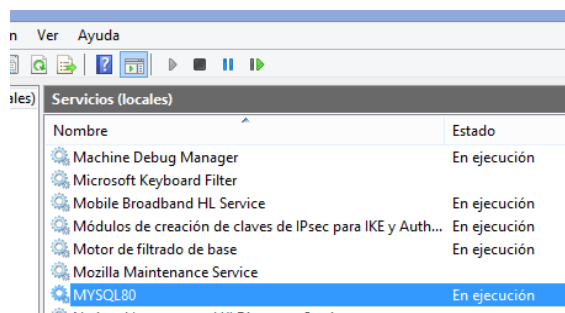
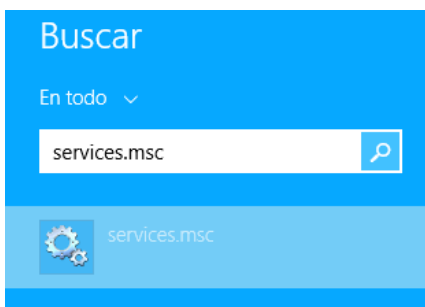
Y luego un error de que no tenemos base de datos.

Crear base de datos con workbench

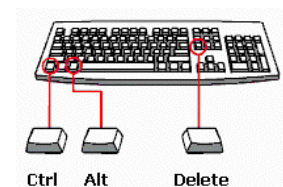
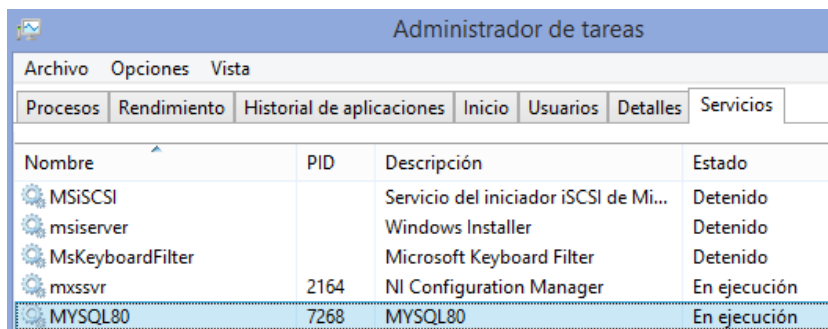
Ejecutamos el software **MySQL Workbench**, utilizado en la materia **Base de Datos**.

Debemos verificar que el servicio MYSQL80 esté en ejecución, de lo contrario no tendremos comunicación con la base de datos que vamos a crear.

Para verificar el estado del servicio podemos buscar : “services.msc” y allí buscamos MYSQL80.

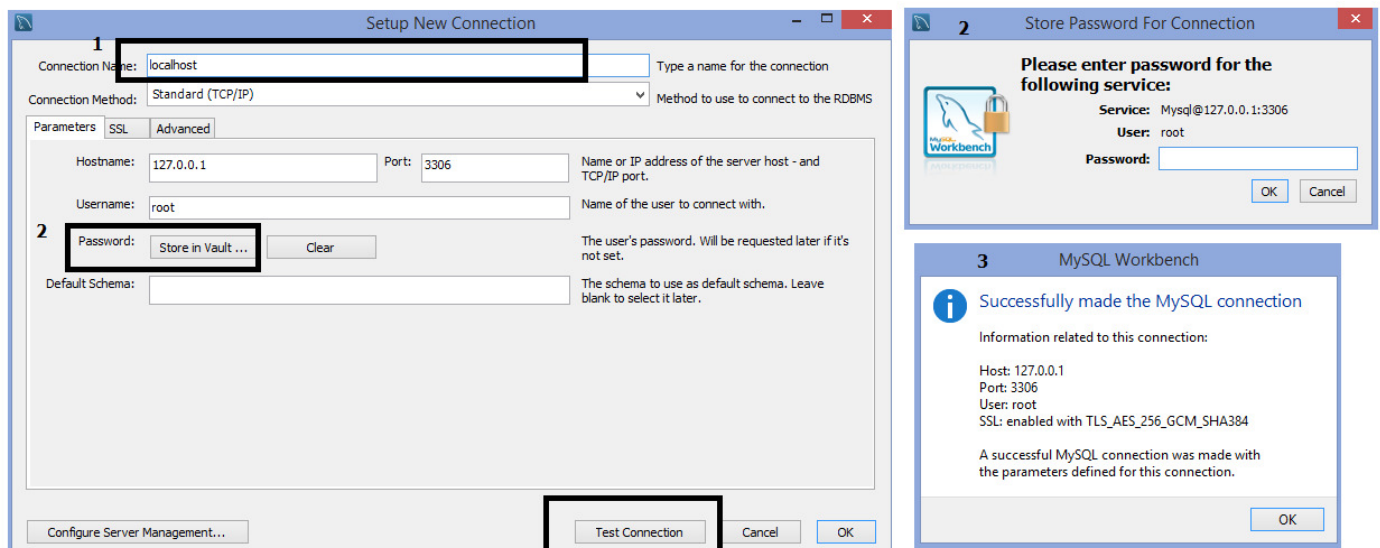


Otra manera es utilizando el comando **Ctrl+Alt+Supr.**



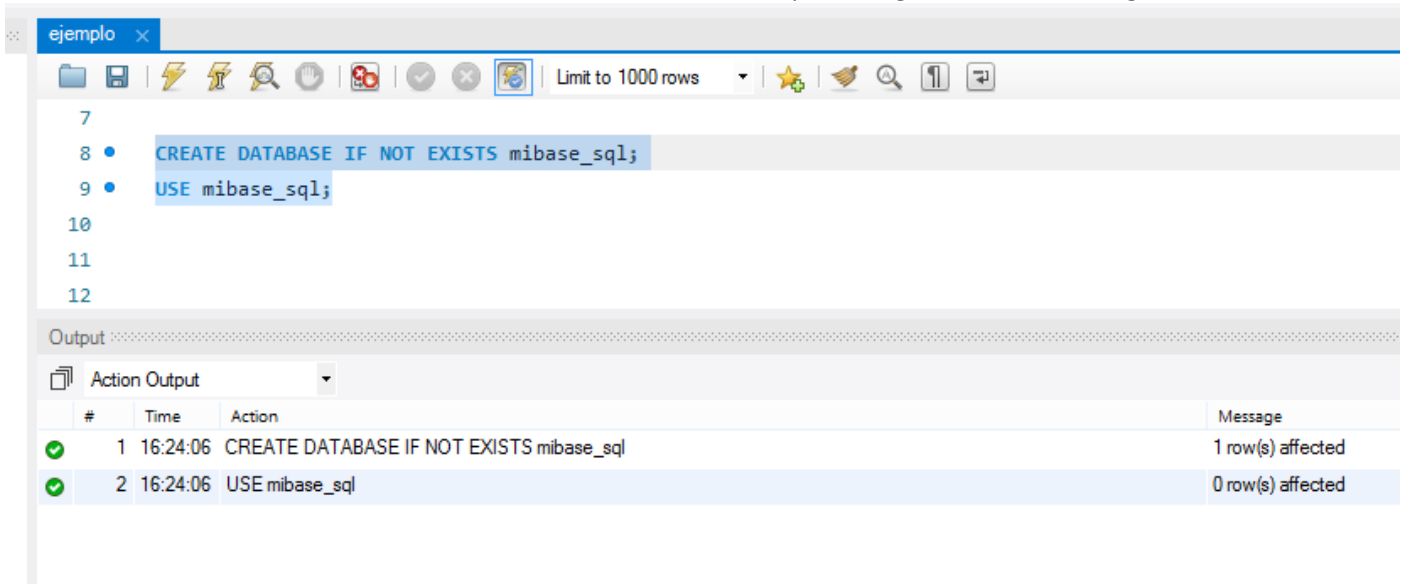
Seleccionar administrador de tareas y clic en la pestaña servicios. Luego buscamos MYSQL80.

Cuando inicia Workbench, en la parte inferior, debemos hacer clic en **+** en MySQL Connections.



En el nombre de la conexión por ejemplo escribimos “localhost”, ingresamos nuestra contraseña configurada en la instalación de Workbench, verificamos el test de connection y OK.

Creamos un nuevo archivo: ejemplo.sql. Dentro del mismo se crea la base de datos: mibase_sql



Ya con esto, y sin desconectarse de Workbench, podemos probar el código que implementamos hasta ahora.

Iniciando el código utilizando la dependencia mysql

```

JS app.js • {} package.json
JS app.js > ...
9   const PORT = 4000;
10
11  // crear constante llamada app y generamos una instancia de express
12  const app = express ();
13
14  // utilizamos app.use y le pasamos bodyparser
15  app.use(express.json());
16
17  // mysql en base a la documentacion:
18  // https://www.npmjs.com/package/mysql
19
20  const connection = mysql.createConnection ({
21    host: 'localhost',

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Windows PowerShell
Copyright (C) 2013 Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo> node app.js
El servidor esta escuchando en el puerto:4000
C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo\node_modules\mysql\lib\protocol\Parser.js:43
7
 throw err; // Rethrow non-MySQL errors
 ^

Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server; consider upgrading MySQL client
 at Handshake.Sequence._packetToError (C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
 at Handshake.ErrorPacket (C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
 at Protocol._parsePacket (C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo\node_modules\mysql\lib\protocol\Protocol.js:291:23)

Lo que se observa es que **no hay** comunicación entre nuestro programa y el workbench.

Para solucionar la falla de conexión, y recordando la clase 9 de programación WEB2, se instaló mysql2.

Con el comando:

```
npm i mysql2
```

solo debemos modificar `const mysql = require ('mysql2');`

Después de la instalación miramos la actualización del package.json

```

package.json
4  "description": "",
5  "main": "app.js",
6  "scripts": {
7    "test": "echo \"Error: no test specified\" && exit 1",
8    "start": "node app.js",
9    "dev": "nodemon --watch app.js"
10 },
11 "keywords": [],
12 "author": "",
13 "license": "ISC",
14 "dependencies": {
15   "mysql": "^2.18.1",
16   "mysql2": "^2.2.5"
17 },
18 "devDependencies": {
19   "body-parser": "^1.19.0",
20   "express": "^4.17.1",
21   "nodemon": "^2.0.7"
22 }
23 }
  
```

```

"dependencies": {
  "mysql": "^2.18.1",
  "mysql2": "^2.2.5"
},
  
```

Iniciando el código utilizando la dependencia mysql2

```

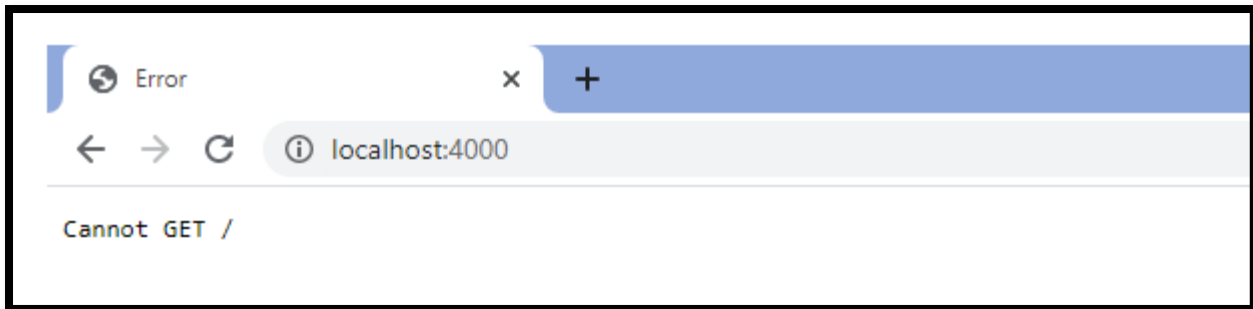
app.js
27  });
28
29  // chequeo de conexión
30  connection.connect(error=>{
31    if(error) throw error;
32    console.info("la base de datos esta funcionando")
33  })
34
35  // escuchar el puerto (alt96 = `)
36  app.listen(PORT,()=>console.info(`El servidor esta escuchando en el puerto:${PORT}`));
37
  
```

```

PS C:\Users\FJTsystems\Desktop\tutorial node_mysql-master\ejemplo> node app.js
El servidor esta escuchando en el puerto:4000
la base de datos esta funcionando
  
```

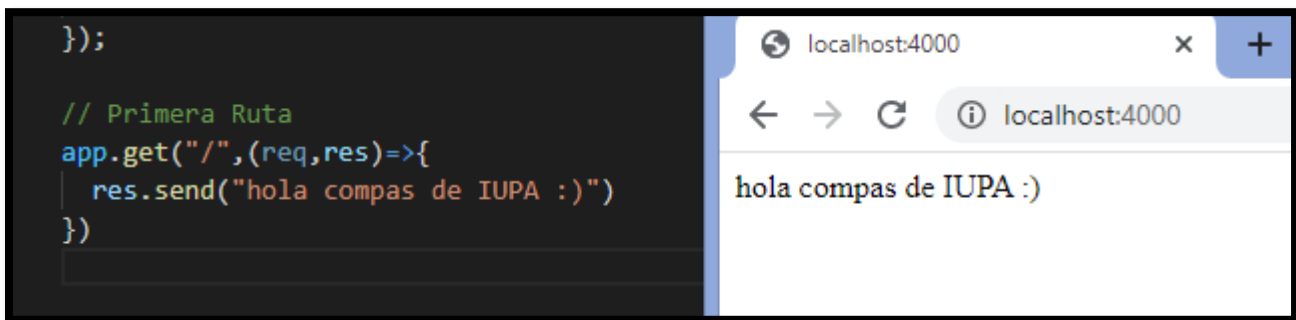
Con mysql2 se conectó correctamente a la base de datos de workbench.

Hasta ahora tenemos la base de datos funcionando, pero si escribimos en el explorador localhost:4000, no hay nada.



A partir de este momento comenzaremos a agregar los endpoints.

Agregamos un método get que envíe un mensaje cuando ingresamos la url: localhost:4000.



Cambio de planes

/// Si nos ponemos a pensar, es mejor utilizar la base de datos utilizada en la materia Base de Datos: **Northwind**

Para ello, solo debemos modificar la configuración de conexión.

```
const connection = mysql.createConnection ({  
  host: 'localhost',  
  user: 'user de workbench',  
  // En el password va tu contraseña de workbench  
  password: 'tu contraseña de Workbench',  
  // database: nombre de la base de datos utilizada en workbench (Materia Base de Datos)  
  database: 'Northwind',  
});
```

Habilitamos el Workbench y trabajamos con los datos de la tabla customers.

```

Database Northwind x clase8
Limit to 1000 rows
21 -- Borra la base Northwind si ya existe.
22 • DROP SCHEMA IF EXISTS Northwind;
23
24 -- Crea la Base de datos: `northwind` y la selecciona como default.
25 • CREATE SCHEMA Northwind;-----
26 • USE Northwind;
27
28 -- Estructura de tabla para la tabla `customers`
29
30 • CREATE TABLE `customers` (
31     `id` int(11) NOT NULL,
32     `company` varchar(50) DEFAULT NULL,
33     `last_name` varchar(50) DEFAULT NULL,
34     `first_name` varchar(50) DEFAULT NULL,
35     `email_address` varchar(50) DEFAULT NULL,
36     `job_title` varchar(50) DEFAULT NULL,
37     `business_phone` varchar(25) DEFAULT NULL,
38     `home_phone` varchar(25) DEFAULT NULL,
39     `mobile_phone` varchar(25) DEFAULT NULL,
40     `fax_number` varchar(25) DEFAULT NULL,

```

Para poder visualizar la tabla debemos implementamos el siguiente código:

```

// leer todos los clientes cuando se ejecuta la url: (get) localhost/4000/customers
app.get ('/customers', (req, res) => {
  connection.query('SELECT * FROM customers', (err, rows, fields) => {
    if(!err) {
      res.json(rows);
    } else {
      console.log(err);
    }
  });
});

```

Código (completo) para pedir todos los campos de la tabla customers.



```

// requerir o importar express
const express = require ('express');
// requerir o importar mysql2
const mysql = require ('mysql2');
// requerir o importar body-parser
const bodyParser = require ('body-parser');

// generar una constante para nuestro puerto
const PORT = 4000;

// crear constante llamada app y generamos una instancia de express
const app = express ();

// utilizamos app.use y le pasamos bodyparser
app.use (express.json ());

// mysql en base a la documentacion:
//https://www.npmjs.com/package/mysql

const connection = mysql.createConnection ({
  host: 'localhost',
  user: 'root',
  // En el password va tu contraseña de workbench
  password: 'root',
  // database: nombre de la base de datos utilizada en workbench (Materia Base de Datos)
  database: 'Northwind',
});

// Primera Ruta
app.get ('/', (req, res) => {
  res.send ('hola compas de IUPA :');
});

// leer todos los clientes
app.get ('/customers', (req, res) => {
  connection.query('SELECT * FROM customers', (err, rows, fields) => {
    if(!err) {
      res.json(rows);
    } else {
      console.log(err);
    }
  });
});

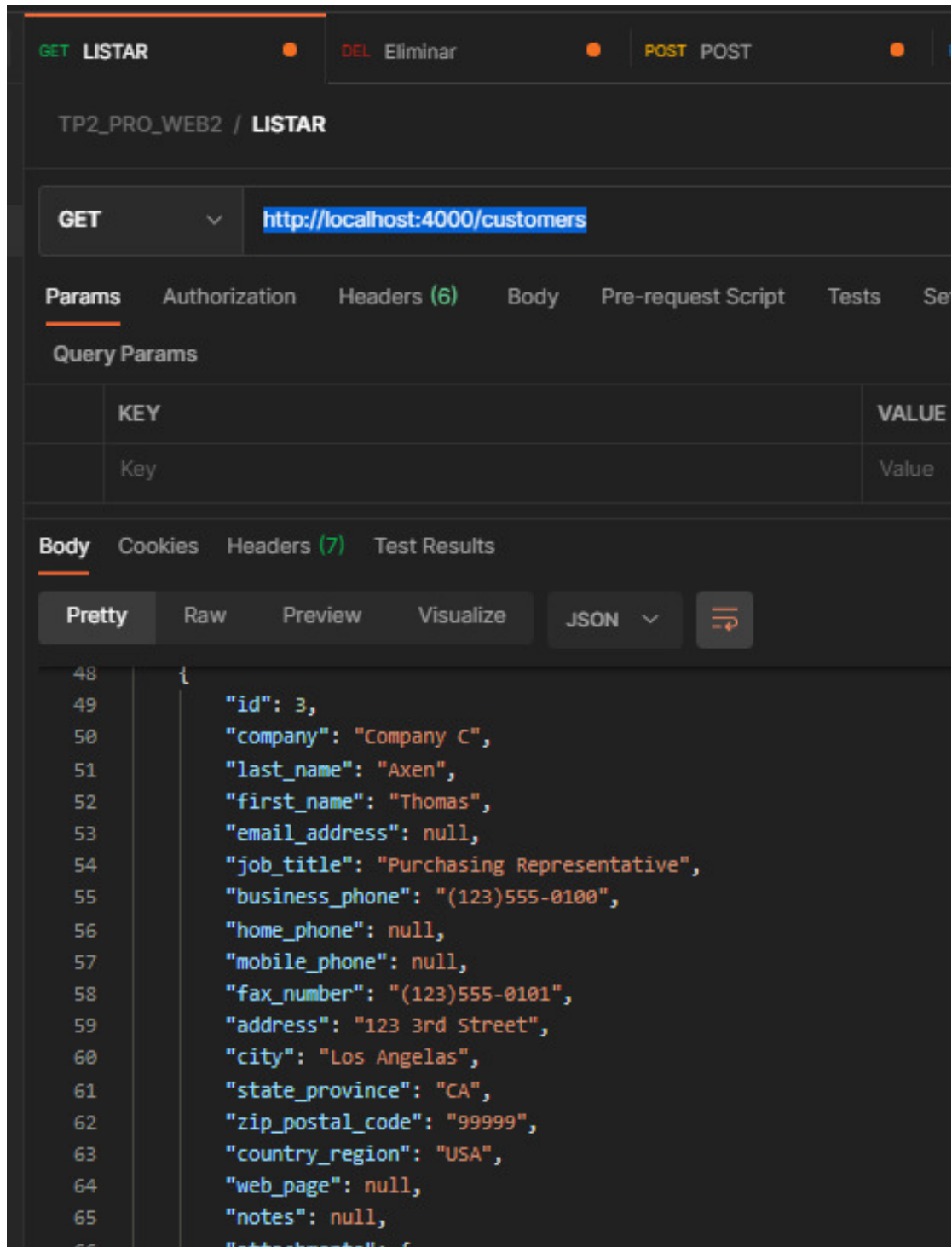
// chequeo de conexión
connection.connect (error => {
  if (error) throw error;
  console.info ('la base de datos esta funcionando');
});

// escuchar el puerto (alt96 = `)
app.listen (PORT, () =>
  console.info (`El servidor esta escuchando en el puerto:${PORT}`)
);

```

Resultado del método get : `http://localhost:4000/customers/`

Utilizando POSTMAN: El servidor devuelve una cantidad de 29 clientes.



Leer un cliente (customers) por id

```
// leer un cliente por id
app.get("/customers/:id",(req,res)=>{
  const{id}= req.params
  const sql= `SELECT * FROM customers WHERE id= ${id}`;
  connection.query(sql, (err, resultado) => {
    if(!err) {
      res.json(resultado);
    } else {
      console.log(err);
      res.send("no hay resultados");
    }
  })
})
```

Resultado del método get : <http://localhost:4000/customers/id>

Al realizar el pedido (utilizando el parámetro id), POSTMAN nos muestra la información del cliente con id=29.

The screenshot shows the Postman interface for a GET request to `http://localhost:4000/customers/id`. The 'Params' tab is active, showing a path variable 'id' with the value '29'. The 'Body' tab is also active, displaying a JSON response in 'Pretty' format. The response is an array containing one object representing a customer.

KEY	VALUE
id	29

```
[
  {
    "id": 29,
    "company": "Company CC",
    "last_name": "Lee",
    "first_name": "Soo Jung",
    "email_address": null,
    "job_title": "Purchasing Manager",
    "business_phone": "(123)555-0100",
    "home_phone": null,
    "mobile_phone": null,
    "fax_number": "(123)555-0101",
    "address": "789 29th Street",
    "city": "Denver",
    "state_province": "CO",
    "zip_postal_code": "99999",
    "country_region": "USA",
    "web_page": null,
    "notes": null,
    "attachments": {
      "type": "Buffer",
      "data": []
    }
  }
]
```

Método POST

En este caso se busca agregar un cliente. Los datos van dentro de un RAW y lo enviamos mediante POSTMAN.

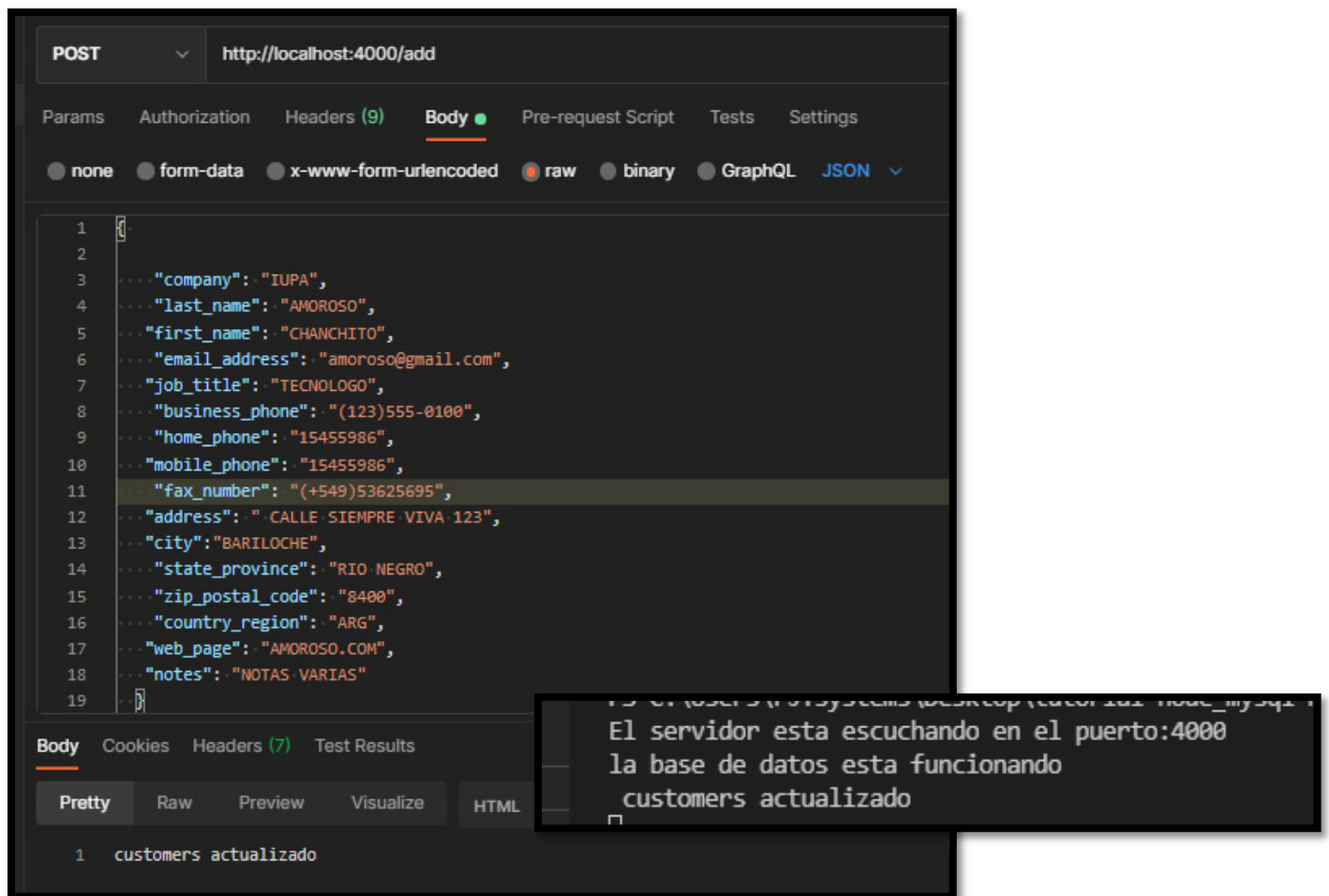
Código para agregar un cliente nuevo con: id=36;

```
// agregar un cliente
app.post('/add', (req, res) => {
  const sql = 'INSERT INTO customers SET ?';
  const customerObject = {
    id: 36,
    company: req.body.company, last_name: req.body.last_name,
    first_name: req.body.first_name, email_address: req.body.email_address,
    job_title: req.body.job_title, business_phone: req.body.business_phone,
    home_phone: req.body.home_phone, mobile_phone: req.body.mobile_phone,
    fax_number: req.body.fax_number, address: req.body.address,
    city: req.body.city, state_province: req.body.state_province,
    zip_postal_code: req.body.zip_postal_code, country_region: req.body.country_region,
    web_page: req.body.web_page, notes: req.body.notes,
  };
  connection.query(sql, customerObject, error => {
    if (error) throw error;
    res.send(" customers actualizado");
    console.log(" customers actualizado");
  })
});
```

Código para enviar en el JSON (raw).

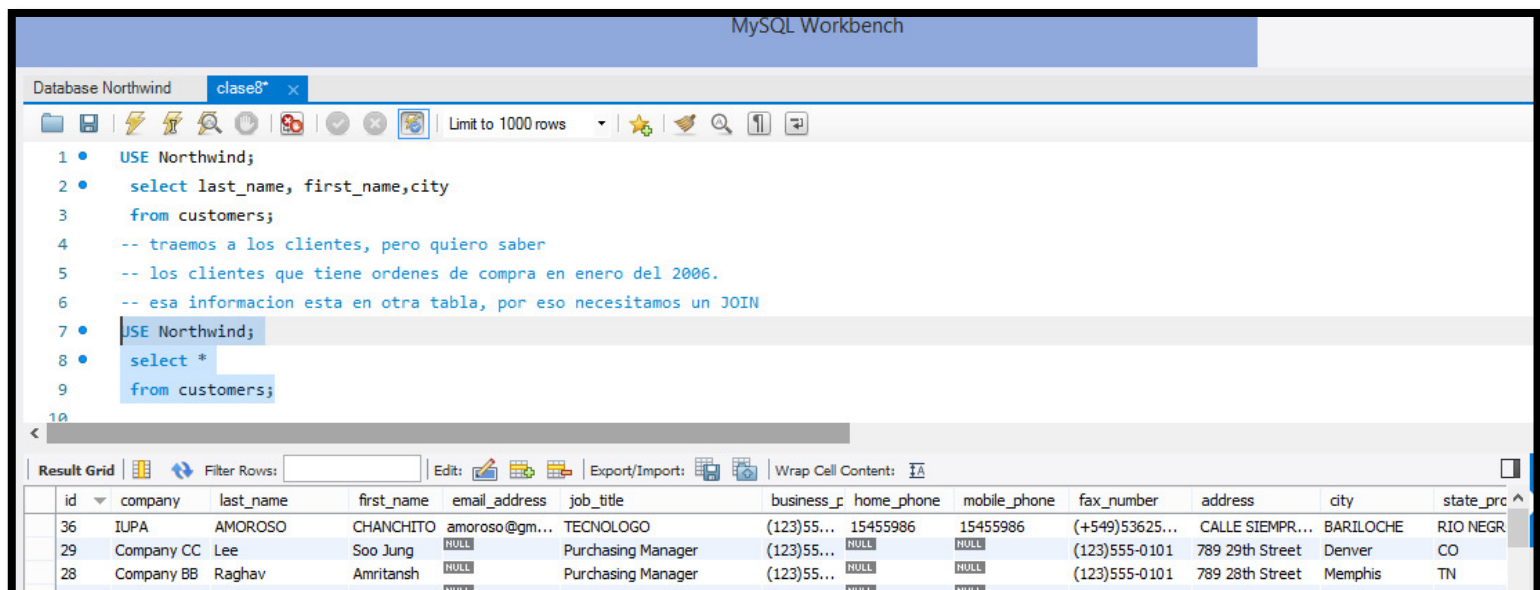
```
{
  "company": "IUPA",
  "last_name": "AMOROSO",
  "first_name": "CHANCHITO",
  "email_address": "amoroso@gmail.com",
  "job_title": "TECNOLOGO",
  "business_phone": "(123)555-0100",
  "home_phone": "15455986",
  "mobile_phone": "15455986",
  "fax_number": "(+549)53625695",
  "address": " CALLE SIEMPRE VIVA 123",
  "city": "BARILOCHE",
  "state_province": "RIO NEGRO",
  "zip_postal_code": "8400",
  "country_region": "ARG",
  "web_page": "AMOROSO.COM",
  "notes": "NOTAS VARIAS"
}
```

Resultado del método POST: `http://localhost:4000/add`



Captura en Visual S.Code

En Workbench se puede observar al cliente nuevo. id=36 (CHANCHITO AMOROSO)

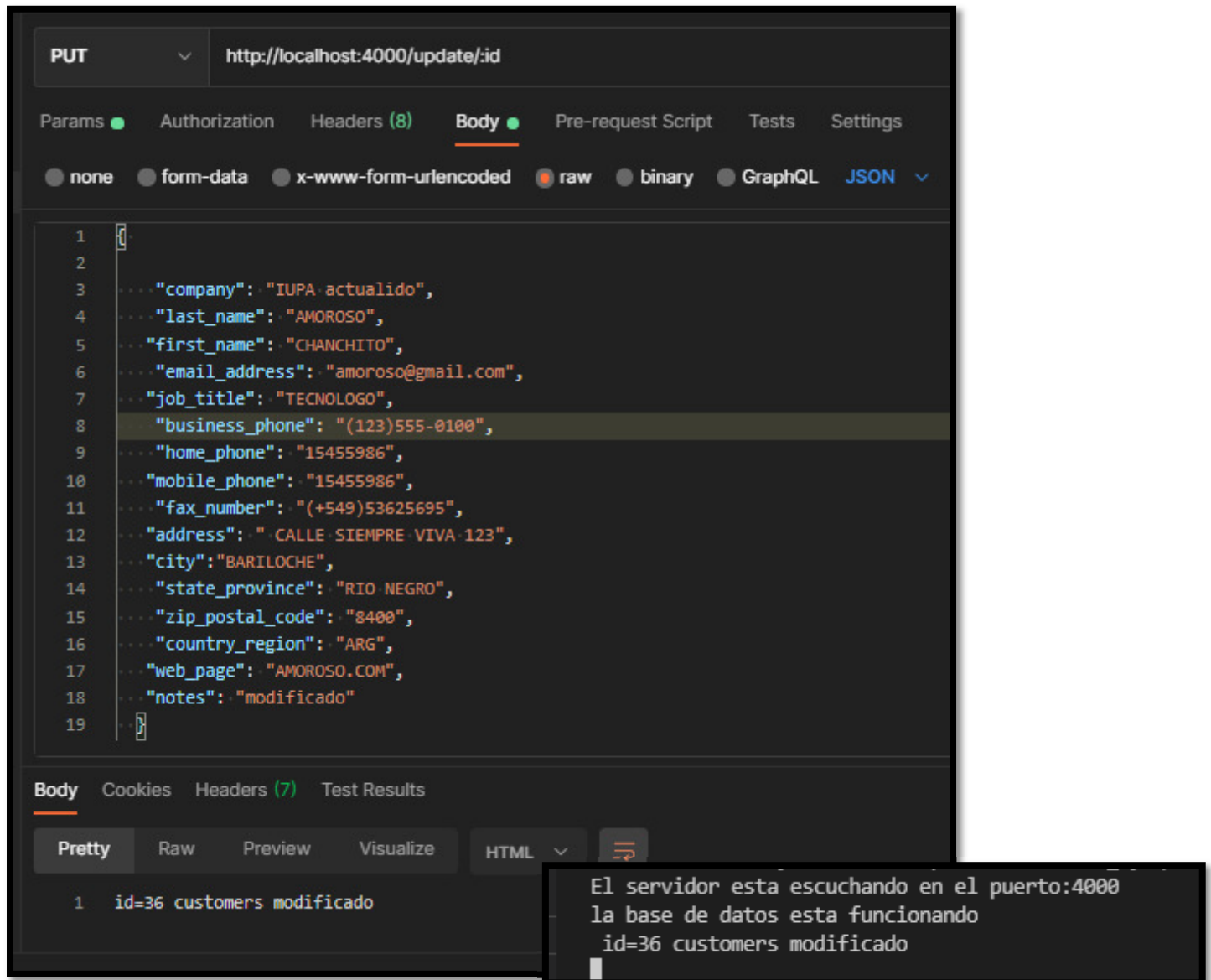


Actualizar id=36 utilizando POSTMAN

Código para actualizar el id=36

```
//actualizar el cliente
app.put ('/update/:id', (req, res) => {
  const {id} = req.params;
  const {
    company,last_name,
    first_name,email_address,
    job_title,business_phone,
    home_phone, mobile_phone,
    fax_number, address,city,
    state_province, zip_postal_code,
    country_region, web_page, notes,
  } = req.body;
  const sql = ` UPDATE customers SET company='${company}',
    last_name='${last_name}',
    first_name='${first_name}',
    email_address='${email_address}',
    job_title='${job_title}',
    business_phone='${business_phone}',
    home_phone='${home_phone}',
    mobile_phone='${mobile_phone}',
    fax_number='${fax_number}',
    address='${address}',
    city='${city}',
    state_province='${state_province}',
    zip_postal_code='${zip_postal_code}',
    country_region='${country_region}',
    web_page='${web_page}',
    notes= '${notes}' where id='${id}'`;
  connection.query (sql, error => {
    if (error) throw error;
    res.send (` id=${id} customers modificado`);
    console.log (` id=${id} customers modificado`);
  });
});
```


Método PUT con parámetro id=36: `http://localhost:4000/update/:id`



Captura en Visual S. Code

Body JSON utilizado en la actualización (parámetro id=36)

```
{
  "company": "IUPA actualido",
  "last_name": "AMOROSO",
  "first_name": "CHANCHITO",
  "email_address": "amoroso@gmail.com",
  "job_title": "TECNOLOGO",
  "business_phone": "(123)555-0100",
  "home_phone": "15455986",
  "mobile_phone": "15455986",
  "fax_number": "(+549)53625695",
  "address": " CALLE SIEMPRE VIVA 123",
  "city": "BARILOCHE",
  "state_province": "RIO NEGRO",
  "zip_postal_code": "8400",
  "country_region": "ARG",
  "web_page": "AMOROSO.COM",
  "notes": "modificado"
}
```

En Workbench se puede observar al cliente con id=36 (CHANCHITO AMOROSO) y una de las modificaciones: “IUPA actualido”

```

8 • select *
9   from customers
10  where id=36;
11

```

Form Editor | Navigate: ⏪ ⏩ 1 / 2 ⏴ ⏵ | Edit: 📄 📋

Id: 36

Company: IUPA actualido

```

7 • USE Northwind;
8 • select *
9   from customers;
10

```

Result Grid | Filter Rows: | Edit: 📄 📋 | Export/Import: 📁 📤 | Wrap Cell Content: ⌵

	id	company	last_name	first_name	email_address	job_title	business_p	home_phone	mobile_phone	fax_number	address	city	state
	36	IUPA actualido	AMOROSO	CHANCHITO	amoroso@gm...	TECNOLOGO	(123)55...	15455986	15455986	(+549)53625...	CALLE SIEMPR...	BARILOCHE	RIO N
▶	29	Company CC	Lee	Soo Jung	NULL	Purchasing Manager	(123)55...	NULL	NULL	(123)555-0101	789 29th Street	Denver	CO

amoroso@gmail.com