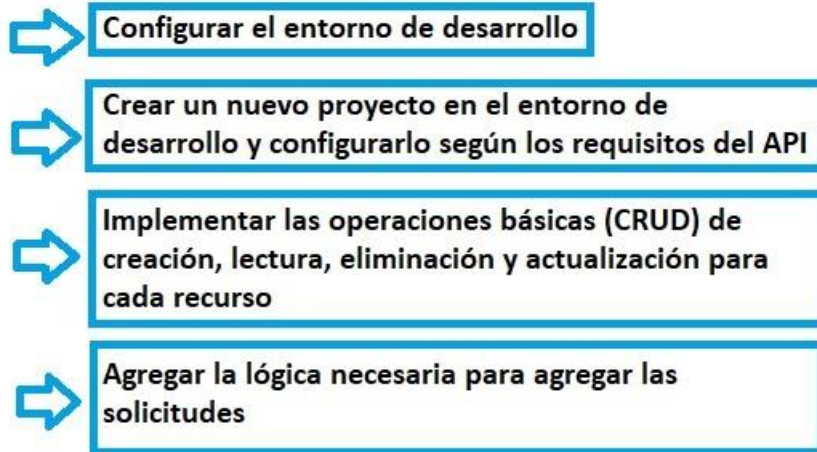


Creación de un api rest desde cero, ejemplo básico

Objetivo: Realizar un api rest básico.



Recursos necesarios:

Es necesario instalar las siguientes aplicaciones o programas:

- Visual studio code versión 1.18.1
- MySQL+MySQL Workbench versión 8.0.32 (MYSQL Community Server–GPL)
- Node.js versión 16.14.1 + npm versión 8.5.0
- Express versión 4.17.1
- Postman versión 10.17.3

Piezas de software necesarias:

- MySQL2 versión 2.2.5
- Nodemon
- Body-parser versión 1.19.2
- Cors versión 2.8.5

Ver [Anexos \(tecnologías y piezas de software\)](#)

Pasos:

1. Configurar el entorno de desarrollo. Ver (Como preparar el entorno de desarrollo).
2. Crear una carpeta para el proyecto, abrir una terminal y ubicarnos en dicho directorio (ver imagen 1).

```
C:\Users\Desarrollador 2\Desktop\desarrollobackend\back>
```

Imagen 1. Ubicación del directorio de proyecto

3. Inicializar el proyecto node.js con el comando "npm init" (ver imagen 2).

```
C:\Users\Desarrollador 2\Desktop\desarrollobackend\back>npm init
```

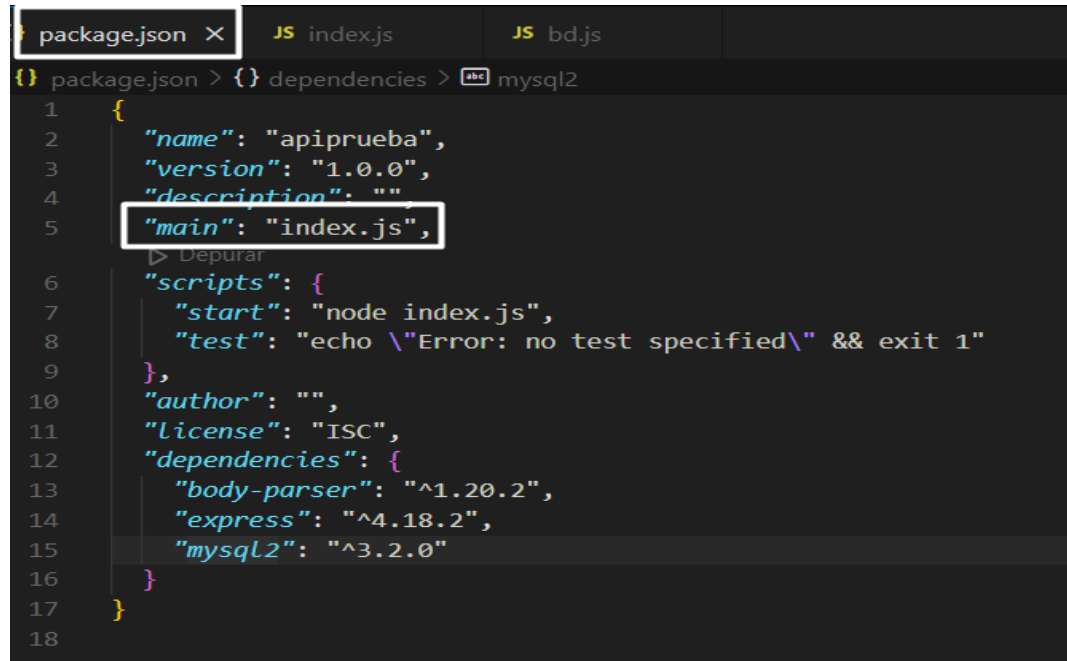
Imagen 2. Comando de inicialización del proyecto en node.js

Nota: Al usar este comando podrás anexar al proyecto (una descripción, repositorio de almacenamiento en git, un keywords y el nombre del autor) si lo considera necesario (ver imagen 3).

```
Press ^C at any time to quit.  
package name: (back)  
version: (1.0.0)  
description:  
entry point: (index.js)  
test command:  
git repository:  
keywords:  
author:  
license: (ISC)
```

Imagen 3. Opciones después de ejecutar el comando "npm init"

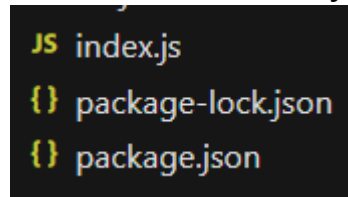
El comando "npm init" permitirá crear el proyecto; automáticamente se creará el **package.json** que es el archivo donde se especifica por defecto el archivo principal index.js (ver imagen 4).



```
package.json X JS index.js JS bd.js
package.json > {} dependencies > mysql2
1 {
2   "name": "apiprueba",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node index.js",
8     "test": "echo \"Error: no test specified\" && exit 1"
9   },
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "body-parser": "^1.20.2",
14    "express": "^4.18.2",
15    "mysql2": "^3.2.0"
16  }
17 }
18
```

Imagen 4. Archivo package.json

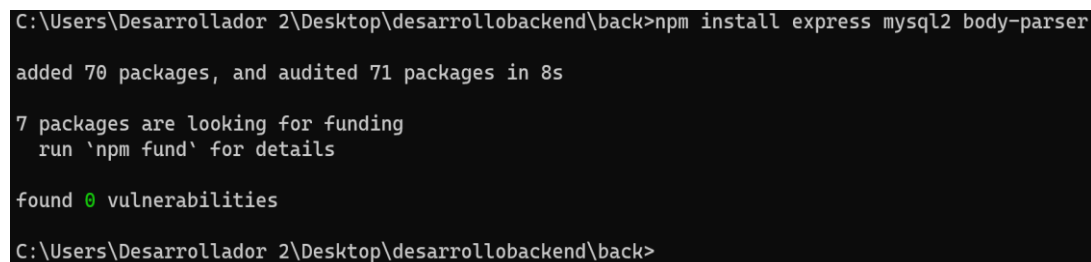
4. Crear el archivo "index.js" en la raíz del proyecto (ver imagen 5).



```
JS index.js
{} package-lock.json
{} package.json
```

Imagen 5. Archivos del proyecto

5. Instalar piezas de software necesarias como: mysql2, body-parse (ver imagen 6).



```
C:\Users\Desarrollador 2\Desktop\desarrollobackend\back>npm install express mysql2 body-parser
added 70 packages, and audited 71 packages in 8s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Desarrollador 2\Desktop\desarrollobackend\back>
```

Imagen 6. Instalación de Piezas de software.

6. Instalar nodemon de manera global. Ver (Como preparar el entorno de desarrollo).

7. Configure la base de datos

- Antes de configurar la base de datos primero verifique que este activo el servicio de MYSQL. Ver (Como iniciar, detener o reiniciar los servicios de MYSQL en Windows).
- Comprobado que el servicio de MYSQL este ejecutándose, abra MySQL Workbench y establezca la conexión con la base de datos (ver imagen 7).

MySQL Connections + ⓘ

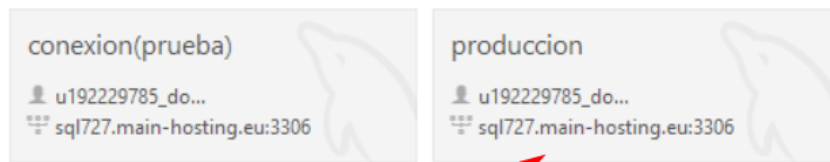


Imagen 7. Conexión con la Base de datos

- Inicie sesión con las credenciales respectivas.



Imagen 8. Credenciales de MySQL

- Una vez que estes conectado a MySQL Workbench, escriba el siguiente comando para crear una base de datos y compile (ver imagen 9).

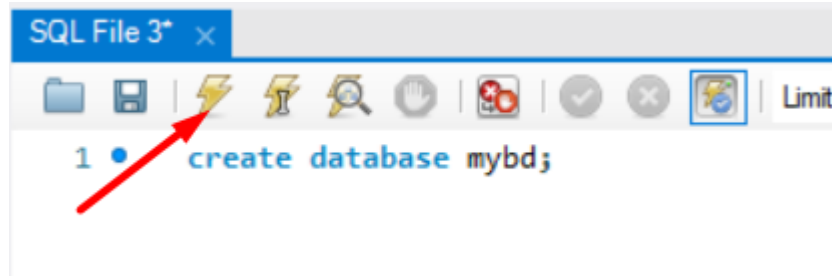


Imagen 9. Comando de creación de base de datos

- Actualice para visualizar la base de datos creada y/o verifique la creación de la Base de datos (ver imagen10).

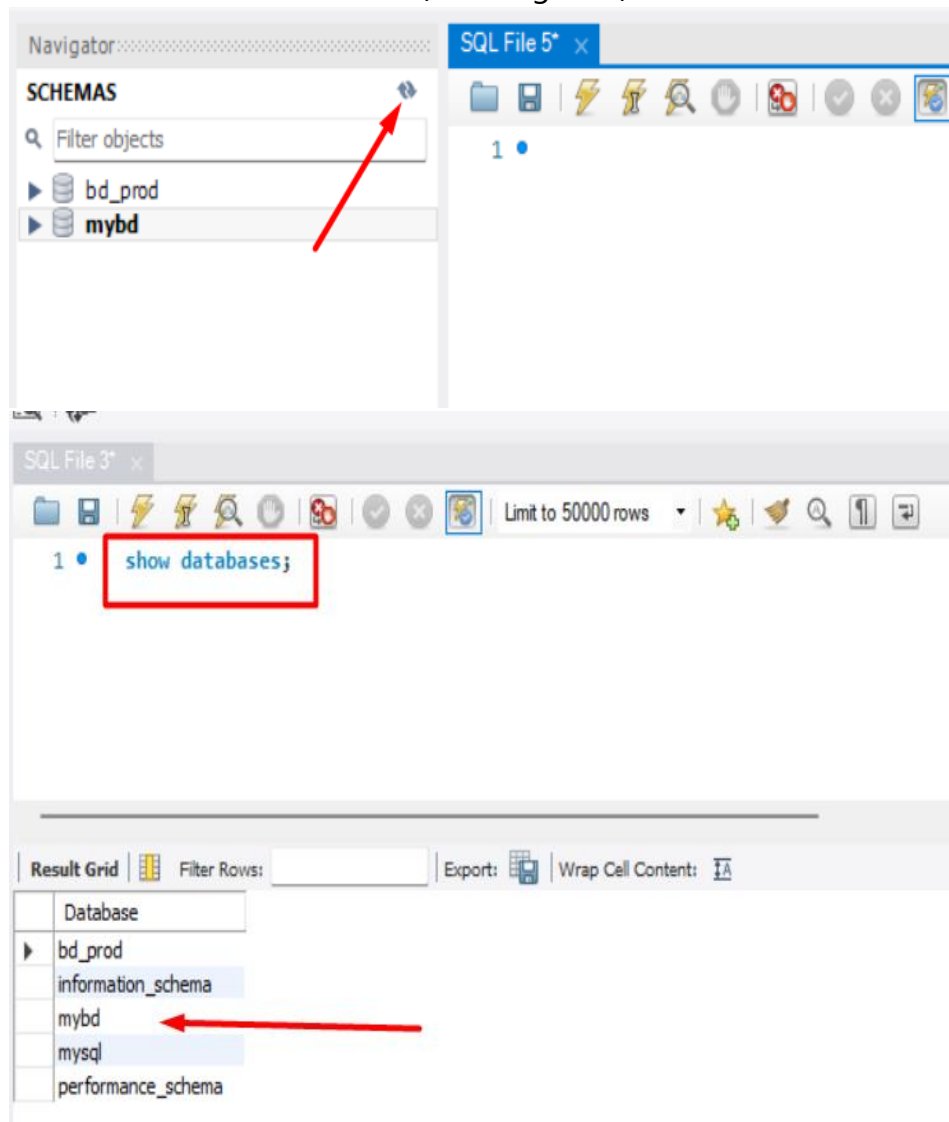


Imagen 10. Comando SQL para ver las bases de datos

- Seleccione la base de datos creada. Para empezar a realizar la estructura de su base de datos (ver imagen 11).

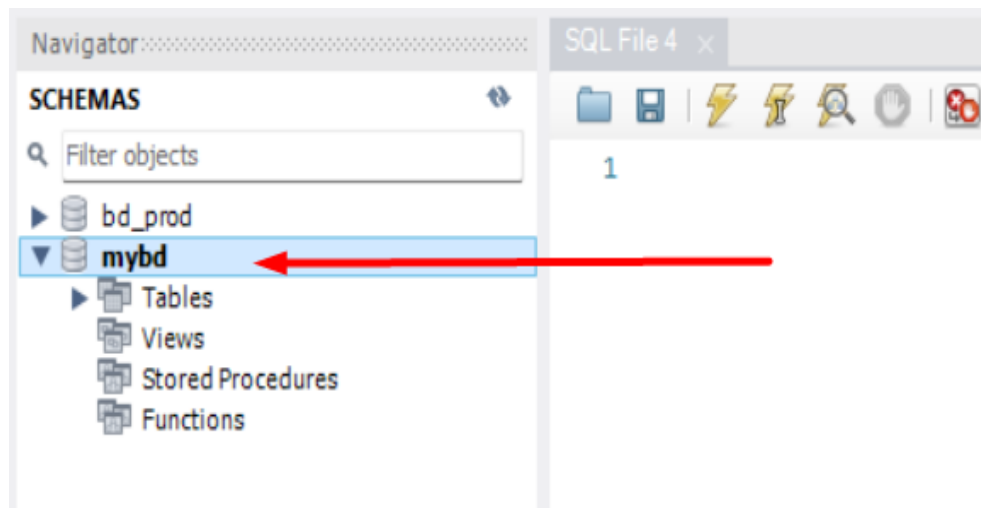


Imagen 11. Selección de base de datos

- Cree una tabla (ver imagen 12).

```
CREATE TABLE usuarios (  
    id INT(11) NOT NULL AUTO_INCREMENT,  
    nombre VARCHAR(50) NOT NULL,  
    email VARCHAR(100) NOT NULL,  
    password VARCHAR(100) NOT NULL,  
    PRIMARY KEY (id)  
);
```

Imagen 12. Secuencia SQL para crear una tabla en la base de datos

- Verifique la creación de la tabla (ver imagen 13).

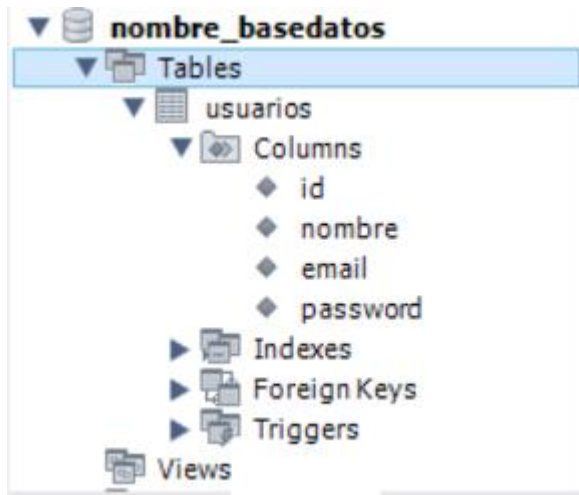


Imagen 13. Visualización de tablas en la base de datos

- Inserte datos a la tabla usuarios para realizar las pruebas necesarias (ver imagen 14).

```
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('josé', 'jose@yopmail.com', '61');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('juan', 'juan@yopmail.com', '62');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('julio', 'julio@yopmail.com', '63');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('jorge', 'jorge@yopmail.com', '64');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('julian', 'julian@yopmail.com', '65');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('jairo', 'jairo@yopmail.com', '66');
INSERT INTO `usuarios` (`nombre`, `email`, `password`) VALUES ('jaime', 'jaime@yopmail.com', '66');
```

Imagen 14. Secuencia SQL para Insertar datos a la tabla usuarios

- Verifique los datos insertados en la tabla usuarios (ver imagen 15).

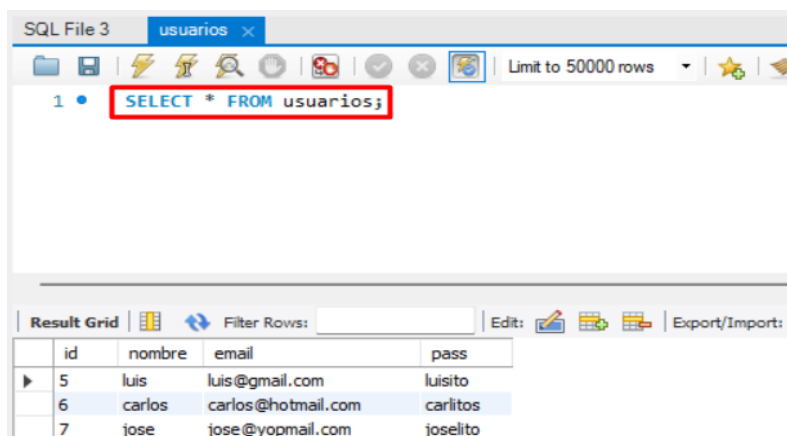


Imagen 15. Secuencia SQL para ver contenido de la tabla usuarios

- Una vez creada la base de datos, se debe verificar que la instalación de MYSQL se ha realizado correctamente en tu proyecto de node.js. Puedes crear un archivo JavaScript de prueba (bd.js) y utilizar el paquete 'mysql' para conectar con la base de datos (ver imagen 16).



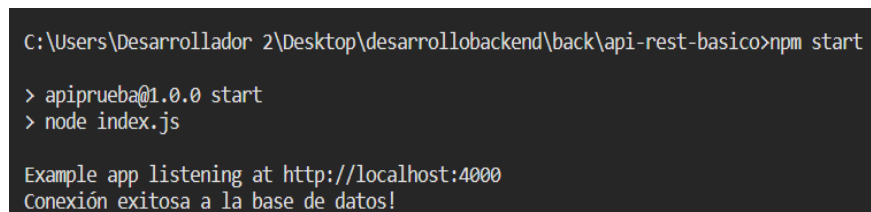
```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'mybd'
});

connection.connect((error) => {
  if (error) {
    console.error('Error al conectarse a la base de datos:', error);
    return;
  }
  console.log('Conexión exitosa a la base de datos!');
});
```

Imagen 16. Verificación de instalación MYSQL en el proyecto

- ¡Comprobar la conexión con la base de datos, ejecute el archivo bd.js que creo para la conexión con la base de datos y verifique el mensaje "conexión exitosa a la base de datos!" (ver imagen 17).



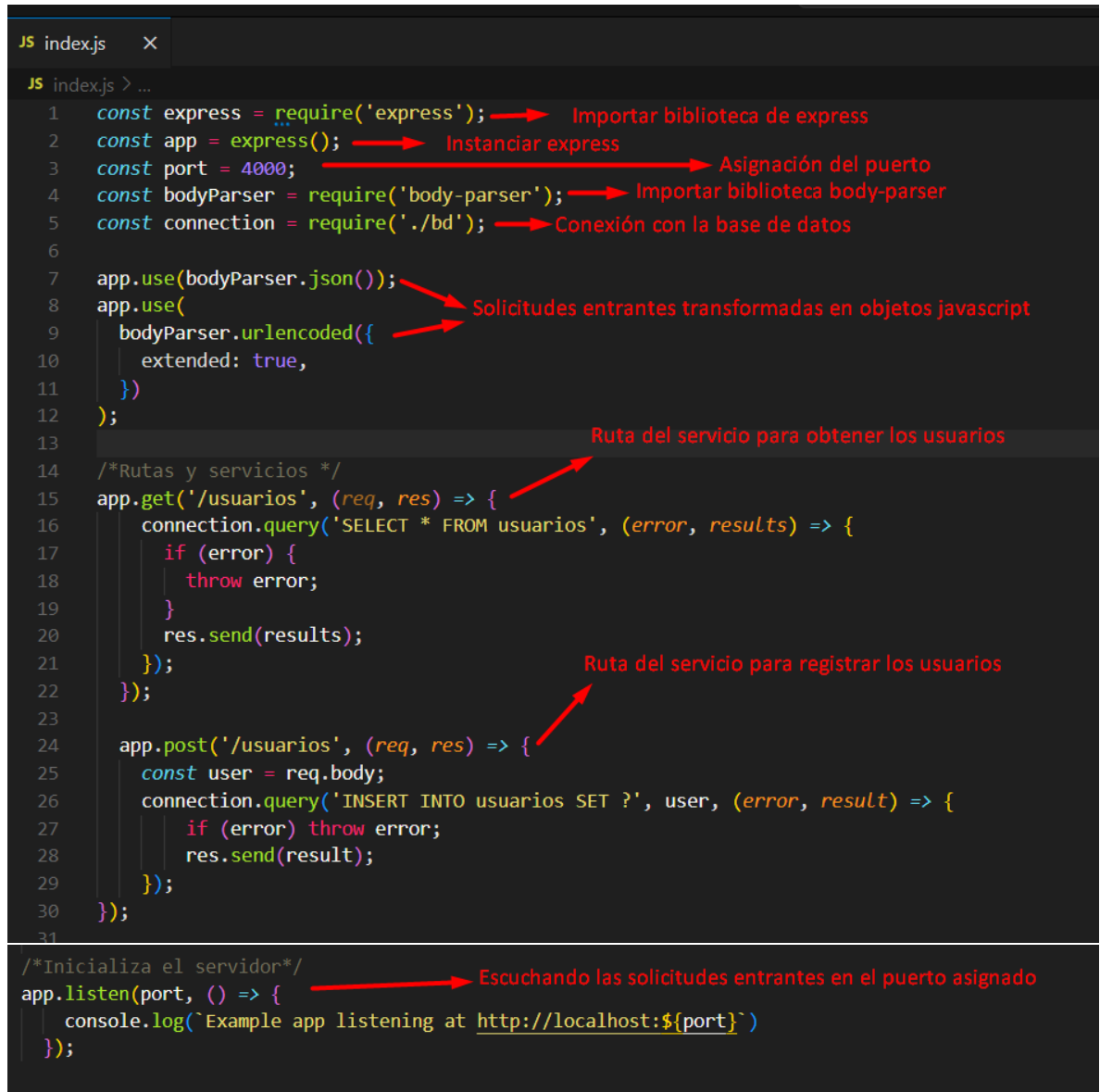
```
C:\Users\Desarrollador 2\Desktop\desarrollobackend\back\api-rest-basico>npm start

> apiprueba@1.0.0 start
> node index.js

Example app listening at http://localhost:4000
Conexión exitosa a la base de datos!
```

Imagen 17. Verificación de conexión con la base de datos

8. Configure el Api Rest con Express (ver imagen 18).



```
JS index.js  X
JS index.js > ...
1  const express = require('express'); // Importar biblioteca de express
2  const app = express(); // Instanciar express
3  const port = 4000; // Asignación del puerto
4  const bodyParser = require('body-parser'); // Importar biblioteca body-parser
5  const connection = require('./bd'); // Conexión con la base de datos
6
7  app.use(bodyParser.json());
8  app.use(
9    bodyParser.urlencoded({ // Solicitudes entrantes transformadas en objetos javascript
10      extended: true,
11    })
12  );
13
14  /*Rutas y servicios */
15  app.get('/usuarios', (req, res) => { // Ruta del servicio para obtener los usuarios
16    connection.query('SELECT * FROM usuarios', (error, results) => {
17      if (error) {
18        throw error;
19      }
20      res.send(results);
21    });
22  });
23
24  app.post('/usuarios', (req, res) => { // Ruta del servicio para registrar los usuarios
25    const user = req.body;
26    connection.query('INSERT INTO usuarios SET ?', user, (error, result) => {
27      if (error) throw error;
28      res.send(result);
29    });
30  });
31
32  /*Inicializa el servidor*/
33  app.listen(port, () => { // Escuchando las solicitudes entrantes en el puerto asignado
34    console.log(`Example app listening at http://localhost:${port}`)
35  });
```

Imagen 18. Descripción del archivo index.js

Nota: Debe tener activo el servicio de MySQL.

9. Suba el servidor con **nodemon** ó **npm start** en la terminal y en Postman realiza las pruebas de los servicios de la api-rest (ver imagen 19).

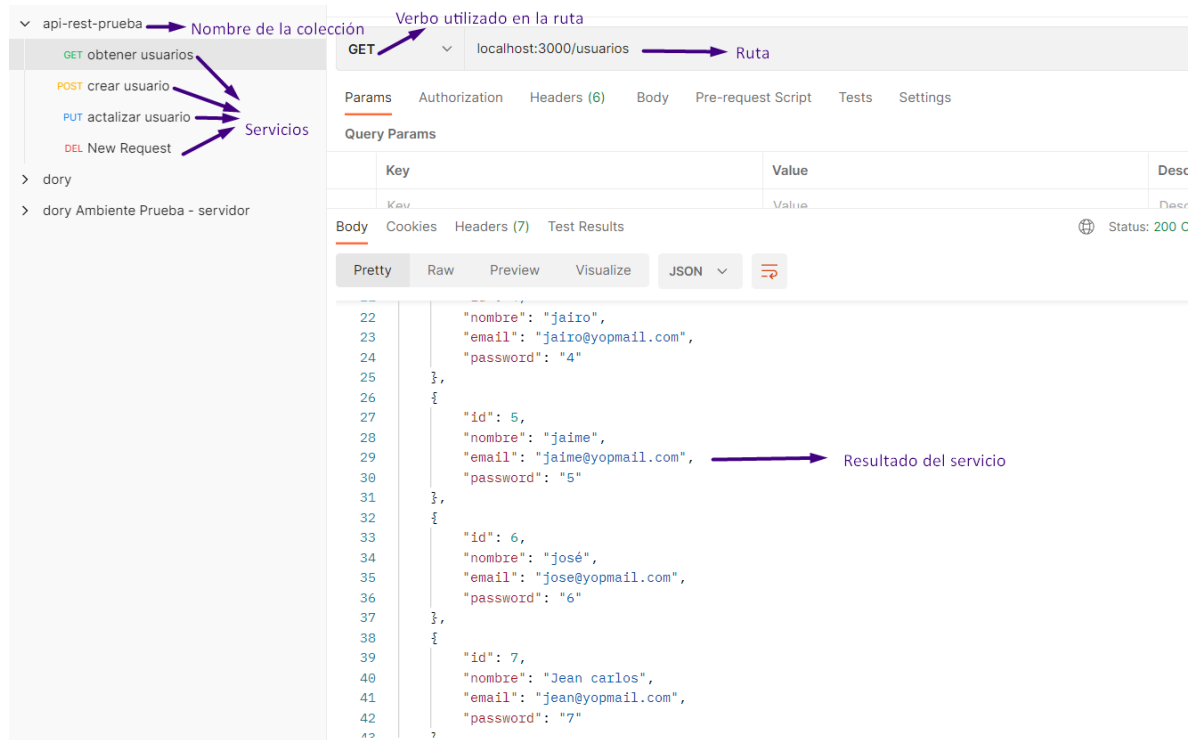


Imagen 19. Descripción de los servicios de usuarios en postman