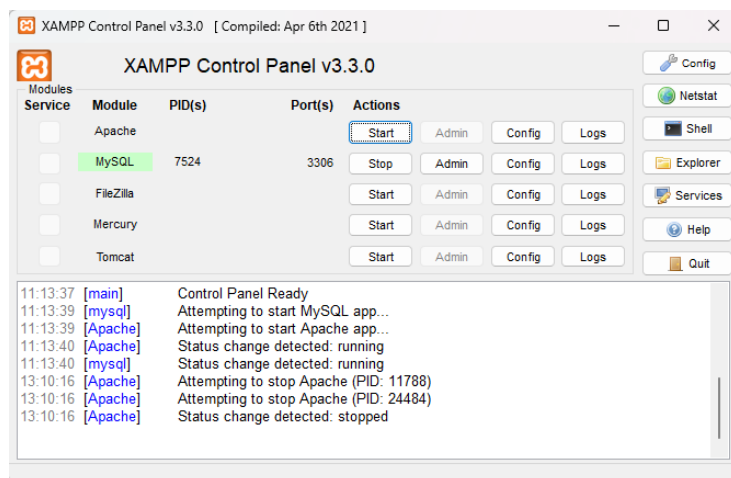


## Manual para el uso de prueba-tecnica-backend-junior

Tener descargado Node.js (de preferencia v20.18.0), igualmente Xamp y Postman o cualquier aplicación que le permita hacer solicitudes https y tener un editor de código.

Clonar el repositorio <https://github.com/EmiValdeCo/prueba-tecnica-backend-junior.git> o descargarlo, si lo ha descargado descomprimir.

Abrir Xamp e iniciar MySQL.



En el gestor de MySql de tu preferencia ejecutar la base de datos que se encuentra en el proyecto con el nombre base\_de\_datos.sql, o aquí te lo proporciono.

```
DROP DATABASE IF EXISTS db_prueba_tecnica;
```

```
CREATE DATABASE db_prueba_tecnica;
```

```
USE db_prueba_tecnica;
```

```
CREATE TABLE tb_usuarios(
```

```
    id_usuario INT AUTO_INCREMENT PRIMARY KEY,
```

```
    nombre_usuario VARCHAR(100) NOT NULL,
```

```
    email_usuario VARCHAR(100) NOT NULL UNIQUE,
```

```
    telefono_usuario VARCHAR(100) NOT NULL UNIQUE,
```

```
    direccion_usuario VARCHAR(255) NOT NULL,
```

```
    usuario VARCHAR(100) NOT NULL UNIQUE,
```

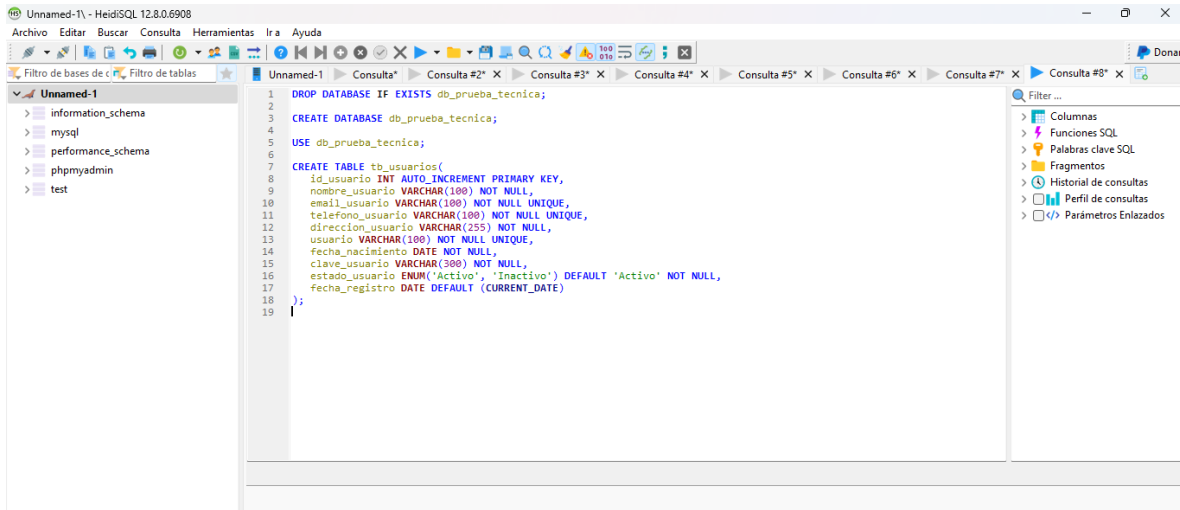
```
    fecha_nacimiento DATE NOT NULL,
```

```
    clave_usuario VARCHAR(300) NOT NULL,
```

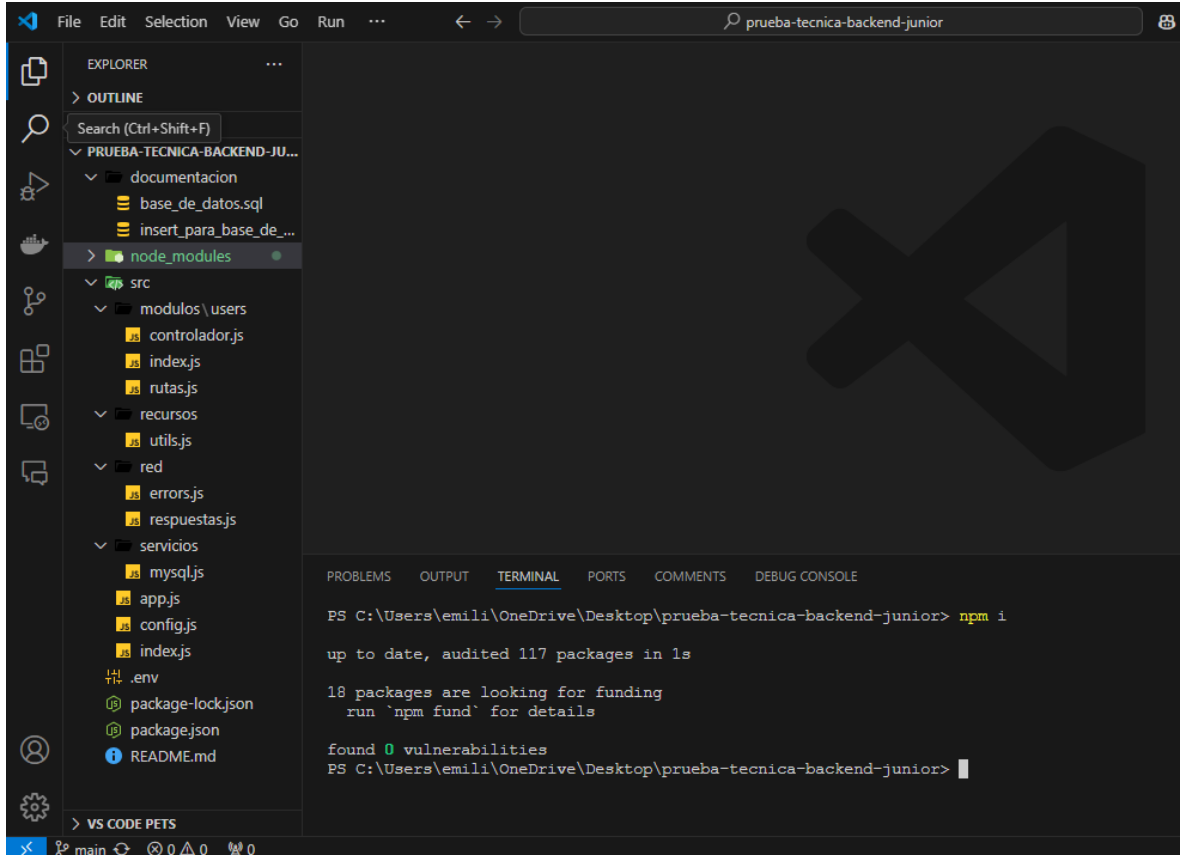
```
    estado_usuario ENUM('Activo', 'Inactivo') DEFAULT 'Activo' NOT NULL,
```

fecha\_registro DATE DEFAULT (CURRENT\_DATE)

);



Abrir el gestor de código de preferencia con el proyecto abierto, abrir la terminal de visual con ctrl + ñ, se tiene que poner el comando npm i, para que se instale las dependencias utilizadas de node.js y funcione correctamente la API.



Inicializar la API poniendo el comando `npm run dev` en la terminal.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  COMMENTS  DEBUG CONSOLE
PS C:\Users\emili1\OneDrive\Desktop\prueba-tecnica-backend-junior> npm run dev
> prueba-tecnica-backend-junior@1.0.0 dev
> nodemon ./src/index.js

[nodemon] 3.1.7
[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`
Servidor escuchado en el puerto 4000
```

Ya inicializada la API, ya se puede hacer las diferentes peticiones de https, abrir tu aplicación de https para comenzar, se puede realizar lo siguiente: Obtener todos los registros de usuarios, obtener solo un usuario, agregar un usuario, actualizar y eliminar.

**Obtener todo:** para obtener todos los usuarios se debe de poner <http://localhost:4000/api/usuarios> y poner en el método GET, y enviarla, donde aparecerá todos los registros de los usuarios.

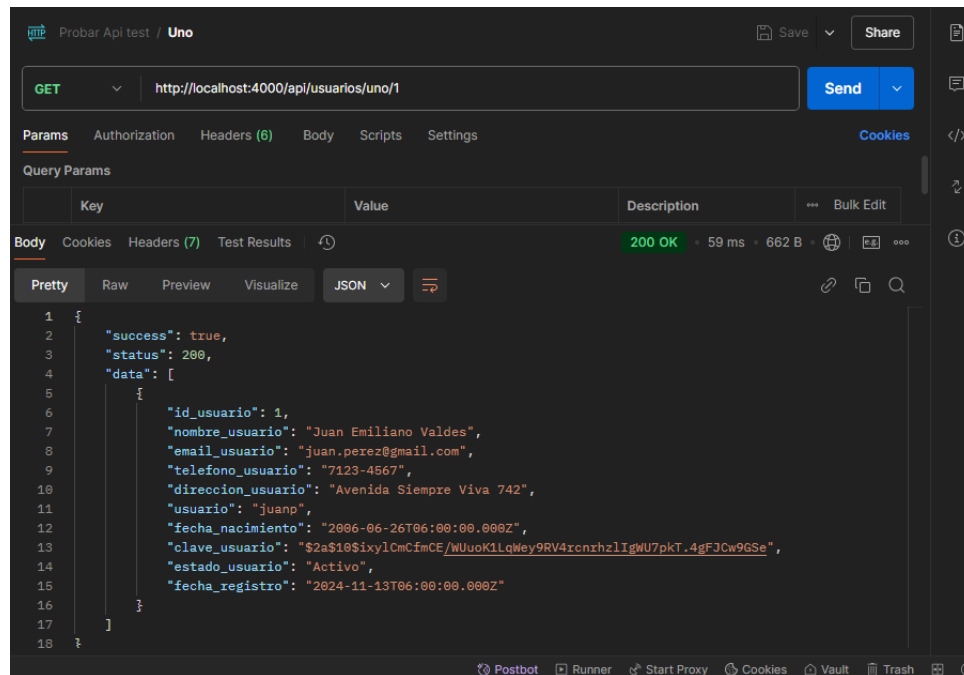
The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** `http://localhost:4000/api/usuarios`
- Status:** 200 OK
- Response Time:** 105 ms
- Response Size:** 1.41 KB

The response body is displayed in JSON format:

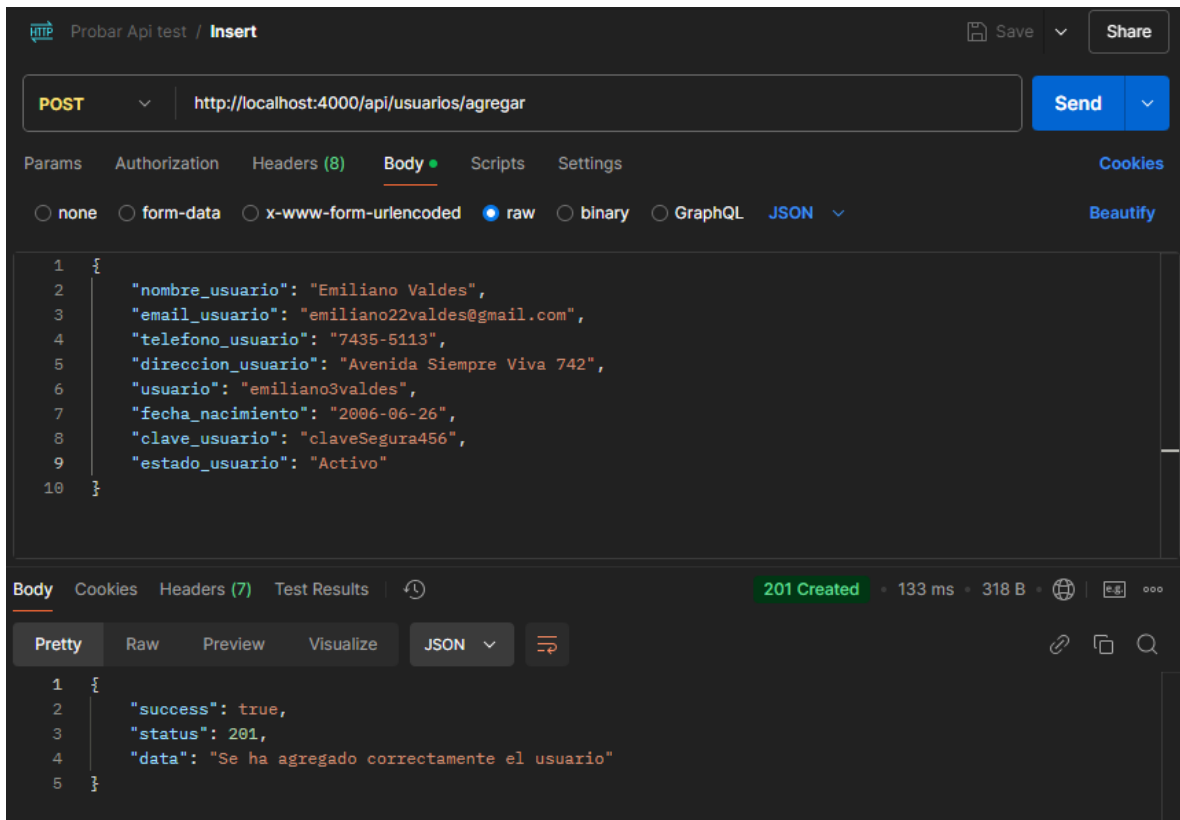
```
{
  "success": true,
  "status": 200,
  "data": [
    {
      "id_usuario": 1,
      "nombre_usuario": "Juan Emiliano Valdes",
      "email_usuario": "juan.perez@gmail.com",
      "telefono_usuario": "7123-4567",
      "direccion_usuario": "Avenida Siempre Viva 742",
      "usuario": "juarp",
      "fecha_nacimiento": "2006-06-26T06:00:00.000Z",
      "clave_usuario": "$2a$10$ixylCmCfmCE/WUuoK1LqWey9RV4rcnrhzlIgwU7pkT.4gFJCw9GSe",
      "estado_usuario": "Activo",
      "fecha_registro": "2024-11-13T06:00:00.000Z"
    }
  ]
}
```

Obtener uno: para obtener un usuario se deberá de colocar la siguiente url donde <http://localhost:4000/api/usuarios/uno/id> se deberá de cambiar el id por un numero de registro del usuario para obtener la información con el método GET.



Agregar usuario: para agregar un usuario se deberá de color la siguiente url <http://localhost:4000/api/usuarios/agregar> donde se debe de poner en el método POST donde se deberá de poner de colocar en el apartado de raw y visualizar correctamente que este seleccionado JSON y este es un ejemplo de cómo debe de colocar para insertar.

```
{
  "nombre_usuario": "Emiliano Valdes",
  "email_usuario": "emiliano22valdes@gmail.com",
  "telefono_usuario": "7435-5113",
  "direccion_usuario": "Avenida Siempre Viva 742",
  "usuario": "emiliano3valdes",
  "fecha_nacimiento": "2006-06-26",
  "clave_usuario": "claveSegura456",
  "estado_usuario": "Activo"
}
```



Actualizar cliente: para actualizar el usuario se debe de colocar la url siguiente <http://localhost:4000/api/usuarios/actualizar> y se debe de poner el método PUT para la realización donde se deberá de poner de colocar en el apartado de raw y visualizar correctamente que este seleccionado JSON y este es un ejemplo de cómo debe de colocar para actualizar.

```
{  
  "id_usuario": "1",  
  "nombre_usuario": "Juan Emiliano Valdes",  
  "email_usuario": "juan.perez@gmail.com",  
  "telefono_usuario": "7123-4567",  
  "direccion_usuario": "Avenida Siempre Viva 742",  
  "usuario": "juanp",  
  "fecha_nacimiento": "2006-06-26",  
  "clave_usuario": "claveSegura456",  
  "estado_usuario": "Activo"  
}
```

The screenshot shows a REST client interface with a PUT request to `http://localhost:4000/api/usuarios/actualizar`. The request body is a JSON object containing user details. The response is a 200 OK status with a JSON body indicating success.

**Request:**

```
PUT http://localhost:4000/api/usuarios/actualizar
```

**Body (JSON):**

```
{  "id_usuario": "1",  "nombre_usuario": "Juan Emiliano Valdes",  "email_usuario": "juan.perez@gmail.com",  "telefono_usuario": "7123-4567",  "direccion_usuario": "Avenida Siempre Viva 742",  "usuario": "juanp",  "fecha_nacimiento": "2006-06-26",}
```

**Response:** 200 OK • 94 ms • 316 B

**Body (JSON):**

```
{  "success": true,  "status": 200,  "data": "Se ha actualizado correctamente el usuario"}
```

Eliminar usuario: para eliminar el cliente se debe de colocar la url siguiente <http://localhost:4000/api/usuarios/eliminar/id> donde se debe de cambiar el id por un número del registro de usuario y se debe de poner el método DELETE para la eliminación del usuario.

The screenshot shows a REST client interface with a DELETE request to `http://localhost:4000/api/usuarios/eliminar/10`. The response is a 200 OK status with a JSON body indicating success.

**Request:**

```
DELETE http://localhost:4000/api/usuarios/eliminar/10
```

**Response:** 200 OK • 8 ms • 314 B

**Body (JSON):**

```
{  "success": true,  "status": 200,  "data": "Se ha eliminado correctamente al usuario"}
```

De igualmente la contraseña sea encuentra encriptado.

```
"id_usuario": 1,  
"nombre_usuario": "Juan Emiliano Valdes",  
"email_usuario": "juan.perez@gmail.com",  
"telefono_usuario": "7123-4567",  
"direccion_usuario": "Avenida Siempre Viva 742",  
"usuario": "juanp",  
"fecha_nacimiento": "2006-06-26T06:00:00.000Z",  
"clave_usuario": "$2a$10$ixylCmCfmCE/WUuoK1LqWey9RV4rcnrhzlIgwU7pkT.4gFJCw9GSe",  
"estado_usuario": "Activo",  
"fecha_registro": "2024-11-13T06:00:00.000Z"
```

De igualmente se encuentra validaciones para la realización donde cuando se equivoca o no cumple con la validación le aparecerá un mensaje indicándole que paso.

```
{  
  "success": false,  
  "status": 401,  
  "error": "Error en la validación",  
  "detalles": [  
    {  
      "type": "field",  
      "value": "holaholahola",  
      "msg": "El correo electrónico debe ser válido",  
      "path": "email_usuario",  
      "location": "body"  
    }  
  ]  
}
```