
SENPAI
ACADEMY

Routers

SENPAI
ACADEMY

Node.js - Routers

Los routers permiten modularizar y reutilizar distintos handlers y middlewares.

Esto es útil para agrupar los manejadores de las rutas, por funcionalidad, o por recurso, etc. Por ejemplo, todas las rutas de Usuario en un Router, todas las de Productos en otro.

En lugar de asignar nuestros manejadores directo contra ***app***, lo hacemos contra un router. Luego le asignamos un path al router como si fuera un manejador de una ruta.

Express provee un middleware en ***express.Router()*** para crear estos manejadores modulares.

```
const express = require('express');
const router = express.Router();
```

SENPAI ACADEMY

Node.js - Routers

```
// En routes/user.js
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  // Devolvemos la lista de usuarios
  res.send({ success: true, users: users });
});
router.get('/:id', function(req, res) {
  const user = users.find(u => u.id === req.params.id);
  if (user) {
    return res.send({ success: true, user: user });
  }
  else {
    return res.send(
      {
        success: false,
        message: `Usuario con Id ${req.params.id} no encontrado`
      }
    );
  }
});

module.exports = router;
```

```
// En index.js => Punto de entrada a nuestro servidor
const express = require('express');
const path = require('path');

const userRoutes =  require('./routes/user');
const productsRoutes =  require('./routes/products');

const app = express();

app.use(express.static(path.join(__dirname, "public")));

//Este middleware se llama cada vez que la app recibe un request
app.use(function (req, res, next) {
  console.log('Hora:', Date.now());
  next();
});

app.use('/user', userRoutes);
app.use('/product', productsRoutes);

app.listen(3000, function () {
  console.log('App corriendo en el puerto 3000');
});
```

SENPAI ACADEMY

Debugging

Node.js - Debugging

Debugging es lo que llamamos al proceso de buscar, encontrar, y solucionar *bugs* en nuestras aplicaciones.

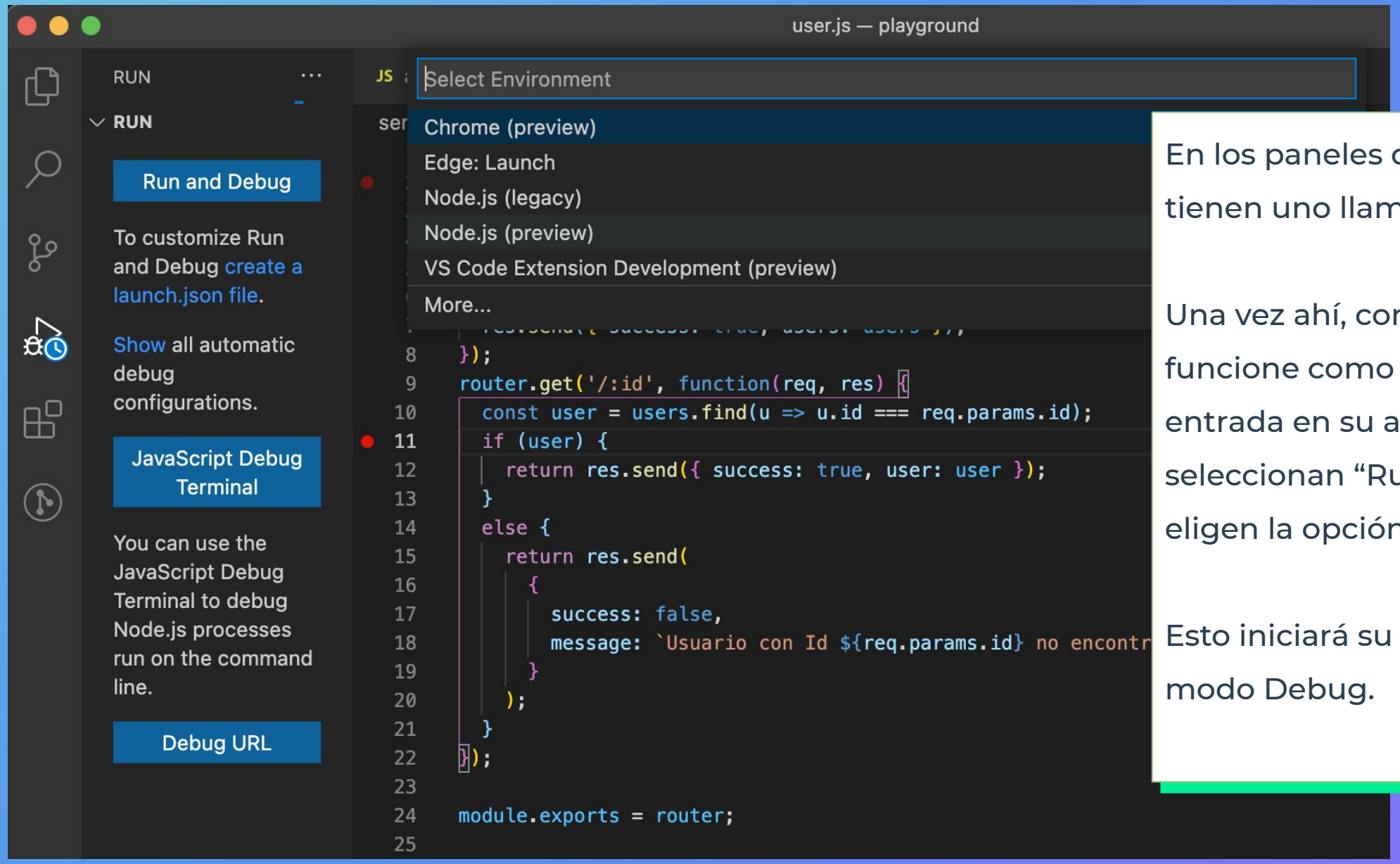
Hay varias formas de realizar esto, pero cuando sabemos que tenemos un error en una ruta del servidor por ejemplo, lo más simple es ejecutar el código línea por línea, y seguir el flujo del código.

Debugging es algo que hacemos en el entorno de desarrollo.

Node.js soporta esto, y si utilizan Visual Studio es aún más simple de realizar.

SENPAI
ACADEMY

Node.js - Debugging



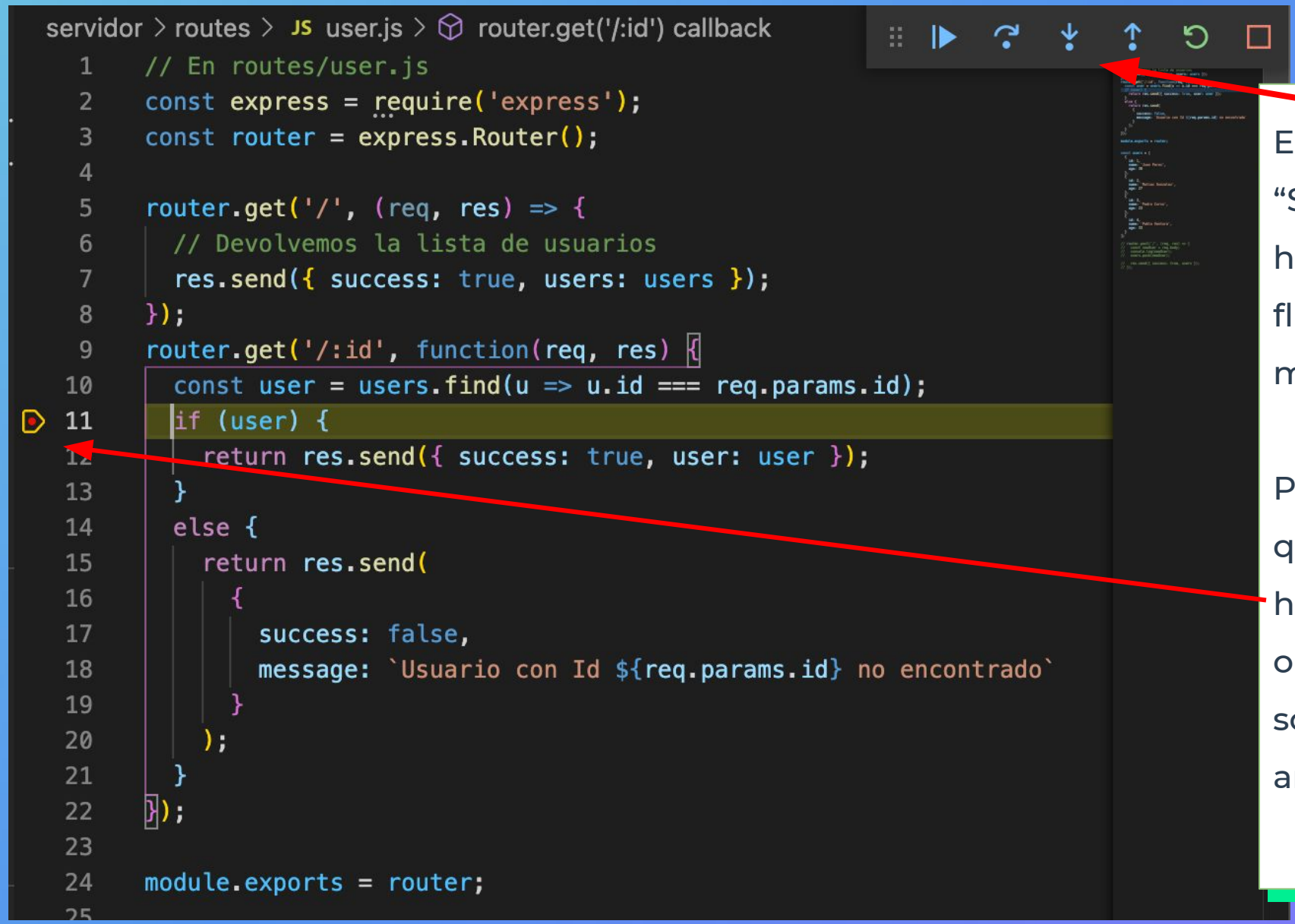
En los paneles de la izquierda, tienen uno llamado “Run”.

Una vez ahí, con el archivo que funcione como punto de entrada en su aplicación, seleccionan “Run and Debug” y eligen la opción “Node.js”.

Esto iniciará su servidor, en modo Debug.

SENPAI ACADEMY

Node.js - Debugging



El panel con el botón de “Play”, “Saltar”, “Entrar”, etc. es su herramienta para controlar el flujo de ejecución una vez que la misma se detenga.

Para indicar en qué línea quiero que se detenga, lo que debo hacer es poner un “BreakPoint” o punto de pausa, clickeando sobre el margen izquierdo del archivo.

SENPAI ACADEMY

Node.js - Debugging

```
2   const express = require('express');
3   const path = require('path');
4
5   const userRoutes =  require('./routes/user');
6   const productsRoutes =  require('./routes/products');
7
8   const app = express();
9
10  app.use(express.static(path.join(__dirname, "public")));
11
12  //Este middleware se llama cada vez que la app recibe un request
13  app.use(function (req, res, next) {
14    console.log('Hora:', Date.now());
15    debugger;
16    next();
17  });
18
```

También podemos indicarle a el Debugger que frene cuando se encuentre con la palabra clave “debugger” en nuestro código.

Esto es algo que agregaremos a nuestro código solo con propósitos de debugging, y luego lo borraremos.

SENPAI ACADEMY

Node.js - Debugging

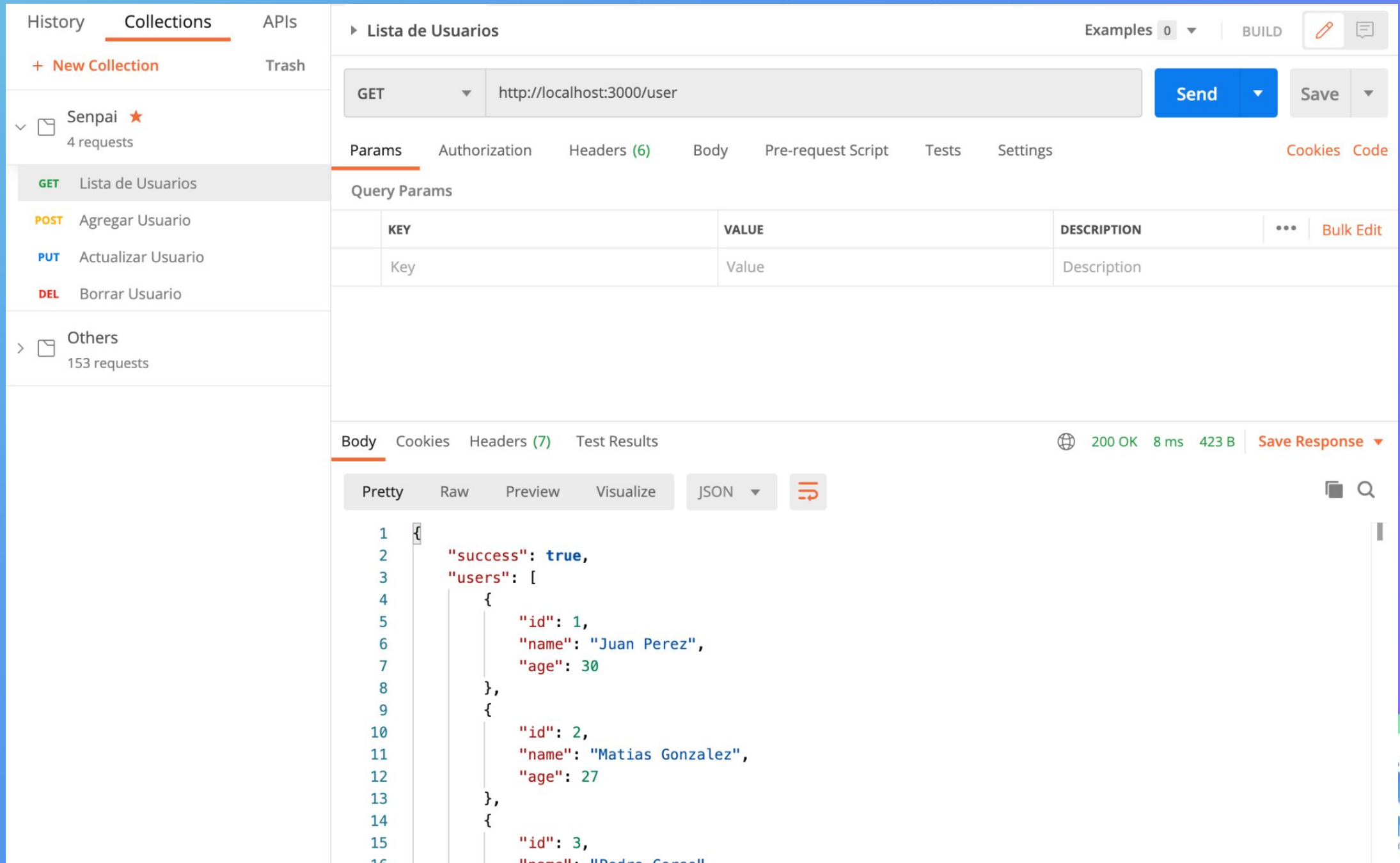
Como vimos, si queremos probar una ruta que responde al verbo **GET**, lo podemos hacer directamente desde el Navegador, pero que pasa si queremos probar un **POST**, **PUT** o **DELETE**?

Para esto podríamos probarlo integrado con nuestro Frontend, pero no siempre vamos a tener eso listo mientras desarrollamos un servidor.

Pero hay otras herramientas que nos van a ser muy valiosas para esto, como Postman! O su prima hermana, Insomnia.

SENPAI ACADEMY

Node.js - Debugging



Node.js - Debugging

En ***Postman*** podemos crear una colección de Requests, cada una a distintas rutas, o a las mismas rutas pero con diferentes parámetros.

Esta herramienta no solo nos sirve para testear nuestro servidor, sino también para simular el uso externo. Recuerden que nuestras **APIs** pueden ser consumidas desde un navegador, pero también desde otro **servidor**, o cualquier aplicación que se comunique con el protocolo **HTTP**, como ***Postman***.

También es muy útil para documentar nuestras **APIs**, ya que la colección que creamos aquí puede contener varios ejemplos de cómo consumir nuestra **API**, y la podemos luego compartir con otros desarrolladores que tengan que acceder a la misma.

SENPAI ACADEMY

SENPAI
ACADEMY

gustavguez

gustavguez

GUSTAVO RODRIGUEZ

FULL STACK DEVELOPER
SOLCRE