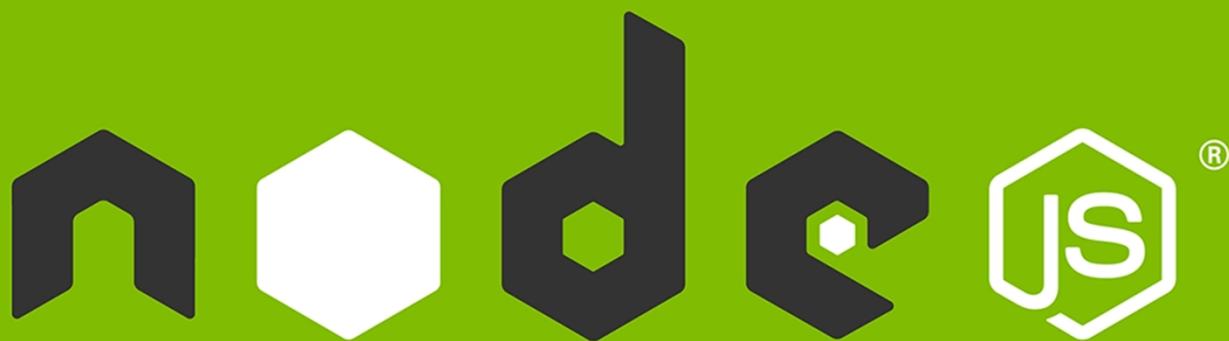


# FRAMEWORK DE DESARROLLO WEB DEL LADO DEL SERVIDOR



# nodejs

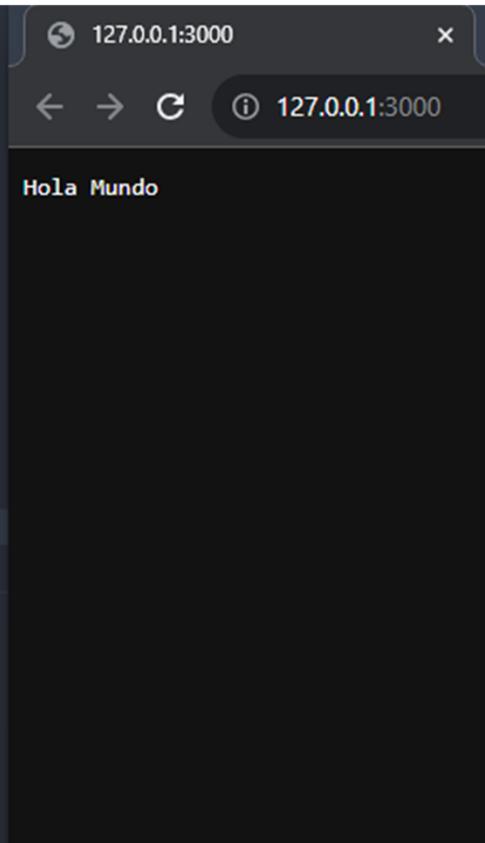
- Es un framework para implementar operaciones de entrada y salida. Está basado en eventos, streams y construido encima del motor de Javascript V8, que es con el que funciona el Javascript de Google Chrome.
- El hola mundo de nodejs, hi.js

```
console.log('Hola Mundo NodeJS !!!');
```

# Versión de nodejs v20.8.0

- node --version

```
[kralos]--[|main ≡ ● ]  
[D:\umar\Asignaturas\Sem  
▶ node --version  
v20.8.0
```



A screenshot of a Node.js application running in a browser. The browser window shows the URL `127.0.0.1:3000` and the response `Hola Mundo`. Below the browser is a code editor showing a file named `server_1.js` with the following content:

```
JS server_1.js > ...
1  const http = require('http');
2
3  const hostname = '127.0.0.1';
4  const port = 3000;
5
6  const server = http.createServer((req, res) => {
7    res.statusCode = 200;
8    res.setHeader('Content-Type', 'text/plain');
9    res.end('Hola Mundo');
10}); <- #6-10 const server = http.createServer
11
12 server.listen(port, hostname, () => {
13   console.log(`El servidor se está ejecutando en `\);
```

The code editor interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. The DEBUG CONSOLE tab is active, showing the terminal output from the Node.js process.

# nodejs

- Nota importante ojo con:



# nodejs

A screenshot of a terminal window in a dark-themed code editor. The window title bar shows tabs for 'JS server\_1.js' and 'JS hi.js'. The active tab contains the following code:

```
JS hi.js
1 console.log('Hola Mundo NodeJS !!!');
```

Below the code, the terminal output is displayed in a scrollable list:

- ► node .\hi.js  
Hola Mundo NodeJS !!!
- ► node hi.js  
Hola Mundo NodeJS !!!
- ► [ ]

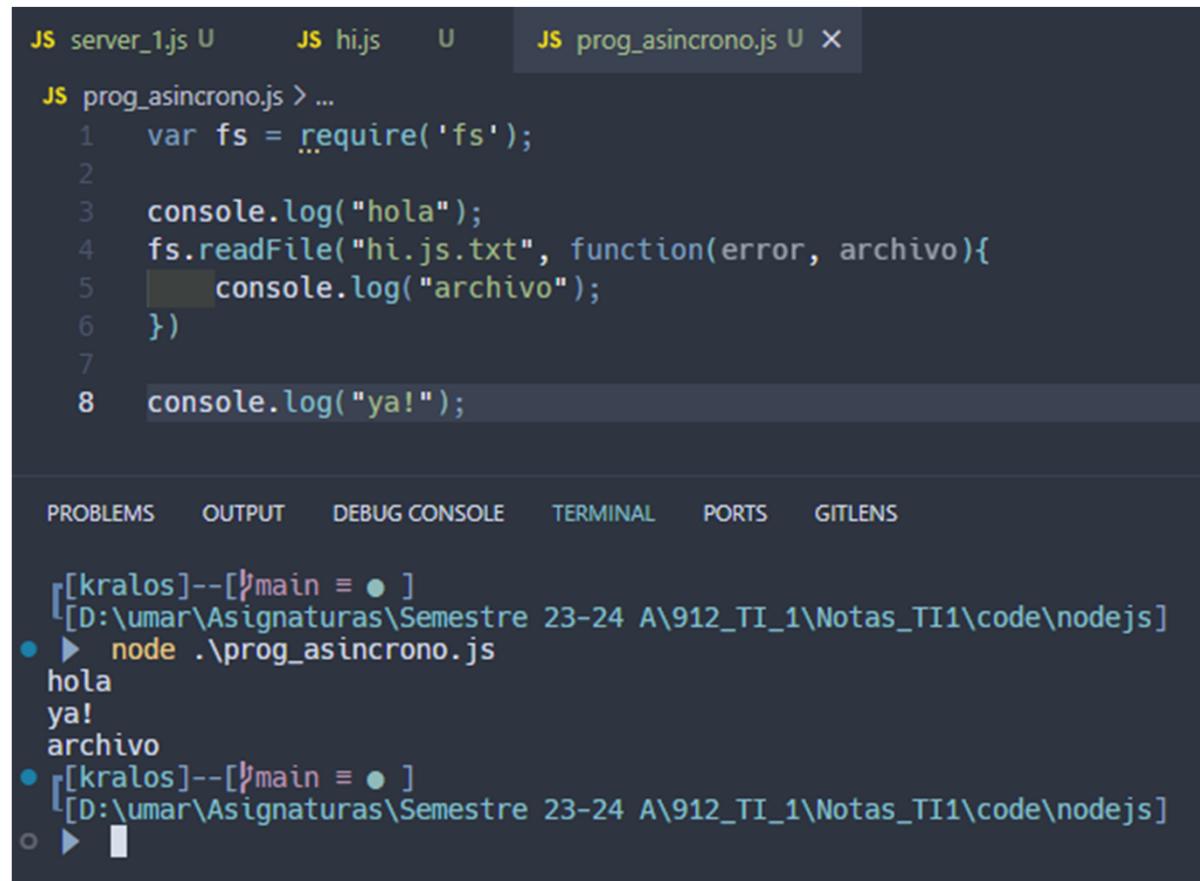
The terminal interface includes a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS.

# Programación Asíncrona

- La filosofía detrás de Node.JS es hacer programas que no bloqueen la línea de ejecución de código con respecto a entradas y salidas, de modo que los ciclos de procesamiento se queden disponibles durante la espera. Por eso todas las APIs de NodeJS usan callbacks de manera intensiva para definir las acciones a ejecutar después de cada operación I/O, que se procesan cuando las entradas o salidas se han completado.

# Programación Asíncrona

- Ejemplo, ver prog\_asincrono.js



```
JS server_1.js U      JS hi.js U      JS prog_asincrono.js U X
JS prog_asincrono.js > ...
1  var fs = require('fs');
2
3  console.log("hola");
4  fs.readFile("hi.js.txt", function(error, archivo){
5    console.log("archivo");
6  )
7
8  console.log("ya!");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
[kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
● ▶ node .\prog_asincrono.js
hola
ya!
archivo
● [kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
○ ▶ █
```

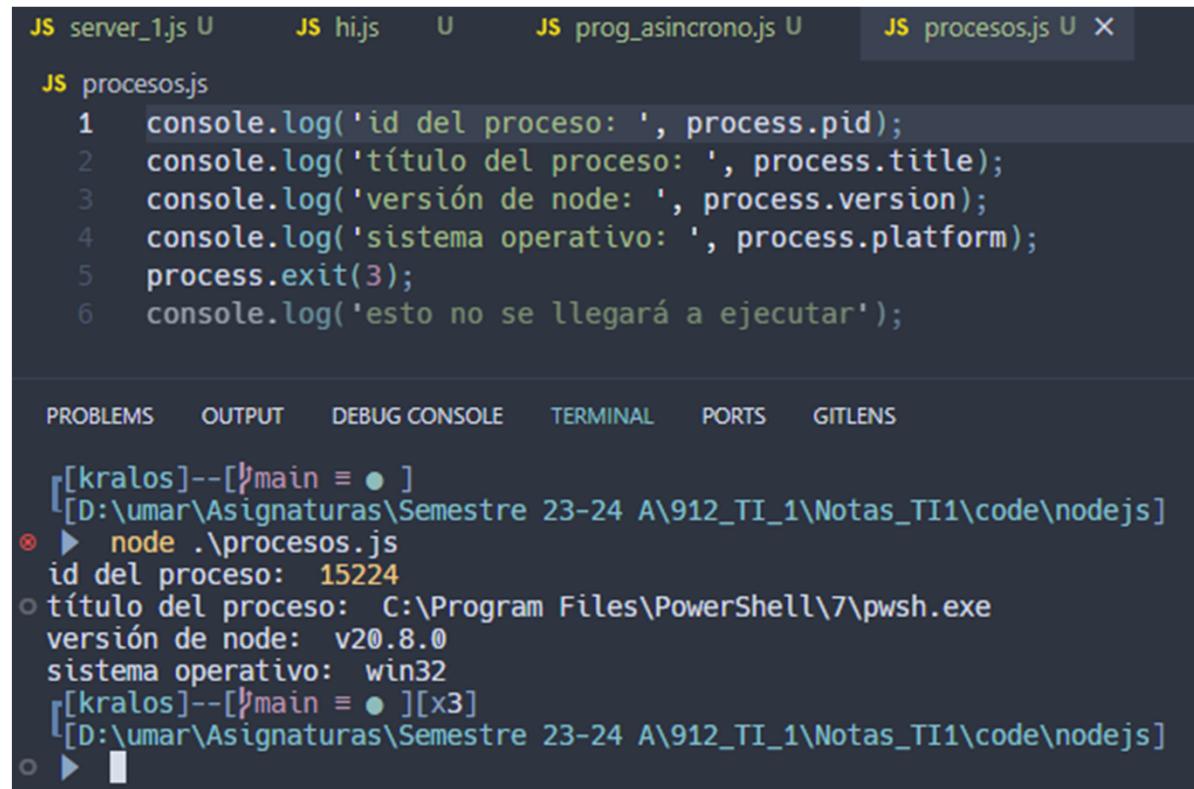
# Programación orientada a eventos (POE)

- En Javascript del lado del cliente tenemos objetos como "window" o "document" pero en Node.JS no existen, pues estamos en el lado del servidor.
- Eventos que podremos captar en el servidor serán diferentes, como "uncaughtError", que se produce cuando se encuentra un error por el cual un proceso ya no pueda continuar. El evento "data" es cuando vienen datos por un stream. El evento "request" sobre un servidor también se puede detectar y ejecutar cosas cuando se produzca ese evento.

# Single Thread (único hilo)

- Una de las características de NodeJS es su naturaleza "Single Thread". Cuando pones en marcha un programa escrito en NodeJS se dispone de un único hilo de ejecución.
- Esto, en contraposición con otros lenguajes de programación como Java, PHP o Ruby, donde cada solicitud se atiende en un nuevo proceso, tiene sus ventajas. Una de ellas es que permite atender mayor demanda con menos recursos, uno de los puntos a favor de NodeJS.

# Single Thread (único hilo)



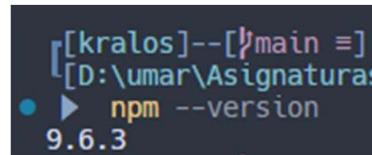
```
JS server_1.js U      JS hi.js U      JS prog_asincrono.js U      JS procesos.js U X
JS procesos.js
1   console.log('id del proceso: ', process.pid);
2   console.log('título del proceso: ', process.title);
3   console.log('versión de node: ', process.version);
4   console.log('sistema operativo: ', process.platform);
5   process.exit(3);
6   console.log('esto no se llegará a ejecutar');

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

[[kralos]--[¶main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
④ ► node .\procesos.js
id del proceso: 15224
○ título del proceso: C:\Program Files\PowerShell\7\pwsh.exe
versión de node: v20.8.0
sistema operativo: win32
[[kralos]--[¶main ≡ ● ][x3]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
○ ► ]
```

# Módulos en NodeJS

- En NodeJS el código se organiza por medio de módulos. Son como los paquetes o librerías de otros lenguajes como Java. Por su parte, NPM es el nombre del gestor de paquetes (package manager) que usamos en Node JS.
- El gestor de paquetes npm al descargar un módulo lo agrega a un proyecto local. Aunque cabe decir que también existe la posibilidad de instalar los paquetes de manera global en nuestro sistema.



```
[kralos]--[?]main ≡]
[D:\umar\Asignaturas
●▶ npm --version
9.6.3
```

# Incluir módulos con "require"

- Javascript nativo no da soporte a los módulos. Esto es algo que se ha agregado en NodeJS y se realiza con la sentencia require(), que está inspirada en la variante propuesta por CommonJS.
- La instrucción require() recibe como parámetro el nombre del paquete que queremos incluir e inicia una búsqueda en el sistema de archivos, en la carpeta "node\_modules" y sus hijos, que contienen todos los módulos que podrían ser requeridos.

```
var http = require("http");
```

# Módulos en NodeJS

- Existen distintos módulos que están disponibles de manera predeterminada en cualquier proyecto NodeJS y que por tanto no necesitamos traernos previamente a local mediante el gestor de paquetes npm.
- Esos toman el nombre de "Módulos nativos" y ejemplos de ellos son "http" para hacer un servidor web y "fs" para el acceso al sistema de archivos.

# Exportando módulos en NodeJS

- Para exportar nuestros propios módulos usamos `module.exports`. Escribimos el código de nuestro módulo, con todas las funciones locales que queramos, luego hacemos un `module.exports = {}` y entre las llaves colocamos todo aquello que queramos exporta.

# Módulos en NodeJS

The screenshot shows a VS Code interface with two files open:

- operaciones.js**:

```
JS operaciones.js > [o] <unknown> > ⚡ multiplicar
1 function suma(a,b){
2     return a + b;
3 }
4
5 function multiplicar(a,b){
6     return a * b;
7 }
8
9 module.exports = {
10     suma: suma,
11     multiplicar: multiplicar
12 }
```
- prueba\_operaciones.js**:

```
JS prueba_operaciones.js > ...
1 var operaciones = require('./operaciones');
2
3 console.log("Sumar ",operaciones.suma(2,3));
4
5 console.log("Multiplicar ",operaciones.multiplicar(2,3));
```

Below the files is a navigation bar with links: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, GITLENS.

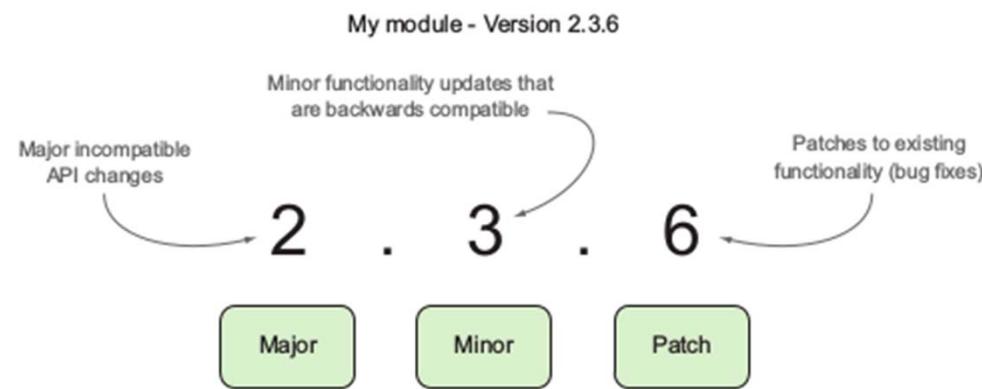
The terminal window at the bottom displays the execution of the script and its output:

```
[[kralos]]--[¶main ≡ ● ]
[[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
● ▶ node .\prueba_operaciones.js
Sumar 5
Multiplicar 6
● [[kralos]]--[¶main ≡ ● ]
○ [[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs]
○ ▶ ]]
```

# Comando npm

- Es un comando que funciona desde la línea de comandos de NodeJS. Por tanto lo tenemos que invocar con npm seguido de la operación que queramos realizar.
  - npm install paquete -g - Instala paquete globalmente.
  - npm install paquete - Instala el paquete de forma local.
  - npm uninstall paquete -g - Desinstala paquete global
  - npm uninstall paquete - Desinstala paquete local

# Módulos en NodeJS



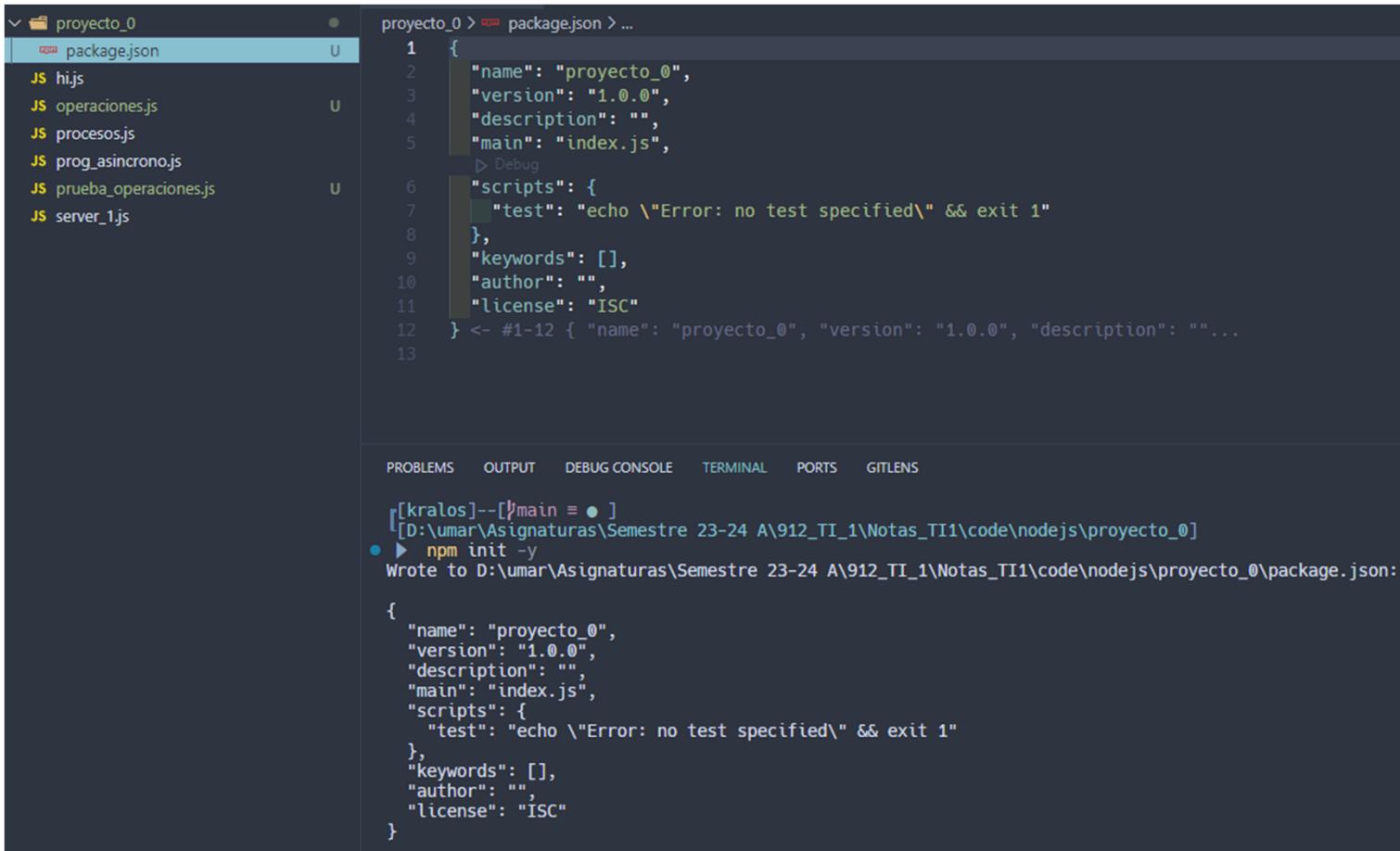
# Comando npm

- `npm install package@0.0.0` Instala una versión específica
- `npm list -g --depth=0` Lista los paquetes globales instalados
- `npm list --depth=0` Lista los paquetes locales instalados
- `npm view paquete versión` Muestra la versión del paquete
- `npm search package` Busca un paquete
- `npm update -g` Actualiza los paquetes globales
- `npm update` Actualiza los paquetes locales
- `npm outdated -g` Lista los paquetes globales anticuados
- `npm outdated` Lista los paquetes locales anticuados
- `npm -v` Muestra la version de npm que esta instalada
- `npm install -g npm-check-updates` instala paquete para chequear actualizaciones

# Otro uso importante de NPM

- Te ayuda a acelerar la fase de desarrollo mediante el uso de dependencias preconstruidas (Crear un archivo package.json).
- npm init
  - El comando init se utiliza para inicializar un proyecto. Crea un archivo package.json.
  - Al ejecutar npm init, se te pedirá que proporciones cierta información sobre el proyecto que estás inicializando.
  - Para saltarte el proceso de proporcionar la información puedes simplemente ejecutar el comando npm init -y.

# Otro uso importante de NPM



The screenshot shows a dark-themed interface of VS Code. On the left, a file tree displays a folder named 'proyecto\_0' containing several JavaScript files: 'hi.js', 'operaciones.js', 'procesos.js', 'prog\_asincrono.js', 'prueba\_operaciones.js', and 'server\_1.js'. The 'package.json' file is selected and open in the main editor area. The code shown is:

```
1 {  
2   "name": "proyecto_0",  
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 } <- #1-12 { "name": "proyecto_0", "version": "1.0.0", "description": "..."  
13
```

Below the editor, a navigation bar includes links for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and GITLENS. The terminal pane at the bottom shows the command 'npm init -y' being run, followed by the message 'Wrote to D:\umar\Asignaturas\Semestre 23-24 A\912\_TI\_1\Notas\_TI1\code\nodejs\proyecto\_0\package.json:' and the contents of the newly generated package.json file.

```
[kralos]--[●]  
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]  
● ► npm init -y  
Wrote to D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0\package.json:  
{  
  "name": "proyecto_0",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}
```

# Otro uso importante de NPM

- Si existe el archivo package.json
- **npm install** Instalar paquetes, descritos en el archivo package.json , de forma global o local.
- **npm uninstall** Desinstalar paquetes, descritos en el archivo package.json , de forma global o local.

# Otro uso importante de NPM

- **npm update**
  - Utiliza este comando para actualizar paquetes a su última versión.
- **npm restart**
  - Se utiliza para reiniciar un paquete o proyecto.
- **npm start**
  - Se utiliza para iniciar un paquete o proyecto.
- **npm stop**
  - Se utiliza para detener la ejecución de un paquete o proyecto.

# ¿Qué son las API?

- Una API es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar los sistemas de software de las aplicaciones. Suele considerarse como el contrato entre el proveedor de información y el usuario, donde se establece el contenido que se necesita por parte del consumidor (la llamada) y el que requiere el productor (la respuesta).

# ¿Qué son las API?

- Las API te permiten interactuar con una computadora o un sistema para obtener datos o ejecutar una función, de manera que el sistema comprenda la solicitud y la cumpla.
- Puedes considerarlas como el mediador entre los usuarios o clientes y los recursos o los servicios web que quieren obtener. Con ellas, las empresas pueden compartir recursos e información mientras conservan la seguridad, el control y la autenticación, lo cual les permite definir los accesos de cada usuario.

# ¿Qué es REST?

- REST no es un protocolo ni un estándar, sino más bien un conjunto de límites de arquitectura. Los desarrolladores de las API pueden implementarlo de distintas maneras.
- Cuando el cliente envía una solicitud a través de una API RESTful, esta transfiere una representación del estado del recurso requerido a quien lo haya solicitado o al extremo. La información se entrega por medio de HTTP en uno de estos formatos: JSON (JavaScript Object Notation), HTML, XML, Python, PHP o texto sin formato.

# ¿Qué es REST?

- JSON es el formato de archivo más popular, ya que tanto las máquinas como las personas lo pueden comprender y no depende de ningún lenguaje, a pesar de que su nombre indique lo contrario.

# ¿Qué es REST?

- Una API RESTful, debe tener:
  - Una arquitectura cliente-servidor compuesta de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP.
  - Una comunicación entre el cliente y el servidor sin estado, lo cual implica que no se almacena la información del cliente entre las solicitudes GET y que cada una de ellas es independiente y está desconectada del resto.
  - Los datos que pueden almacenarse en caché y optimizan las interacciones entre el cliente y el servidor.

# Express

- Es el framework más usado en NodeJS, que nos facilitará la creación de servidores web en muy pocos minutos, personalizados según nuestras necesidades.
- Permitirá no solo servir archivos estáticos, sino atender todo tipo de solicitudes complejas que impliquen realizar cualquier tipo de acción, configurar rutas de manera potente, trabajar de detalladamente con las cabeceras del HTTP y las respuestas, etc. Incluso desarrollar tu propio API REST de una manera más o menos sencilla.

# Express

The screenshot shows the VS Code interface with the following details:

- EXPLORER: PROYECTO\_0**: Shows files: node\_modules (green dot), package-lock.json (blue U), and package.json (orange M).
- package.json M X**: The file content is as follows:

```
1 {   You, 3 days ago • update notas ...
2   "name": "proyecto_0",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   ▶ Debug
7   "scripts": {
8     "test": "echo \\\"Error: no test specified\\\" && exit 1"
9   },
10  "keywords": [],
11  "author": "",
12  "license": "ISC",
13  "dependencies": {
14    "express": "^4.18.2"
15  }
15 } <- #1-12 { "name": "proyecto_0", "version": "1.0.0", "description": ""}.
```

- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, **TERMINAL**, **PORTS**, **GITLENS** tabs are visible at the bottom.
- TERMINAL** tab content:

```
[kralos]--[ψmain ≡]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
● ▶ npm install express

added 58 packages, and audited 59 packages in 9s

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

found 0 vulnerabilities
[kralos]--[ψmain ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
```

# Express

The screenshot shows the VS Code interface with the following details:

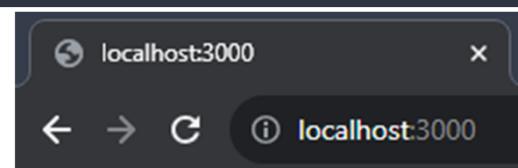
- EXPLORER: PROYECTO\_0**: Shows files: node\_modules, app.js (selected), package-lock.json, packagejson.
- PACKAGE JSON**: Shows the contents of package.json.
- JS app.js**: Shows the code for app.js:

```
const express = require('express');
const app = express();
const hostname = '127.0.0.1';
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!')
});

app.listen(port, hostname, () => {
  console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`)
});
```
- PROBLEMS**: Shows no problems.
- OUTPUT**: Shows the terminal output:

```
[kralos]--[|]main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
▶ node .\app.js
● El servidor se está ejecutando en http://127.0.0.1:3000/
```
- DEBUG CONSOLE**: Shows the URL: localhost:3000.
- TERMINAL**: Shows the command: node .\app.js and the output: El servidor se está ejecutando en http://127.0.0.1:3000/.
- PORTS**: Shows no ports.
- GITLENS**: Shows no lenses.



Hello World!

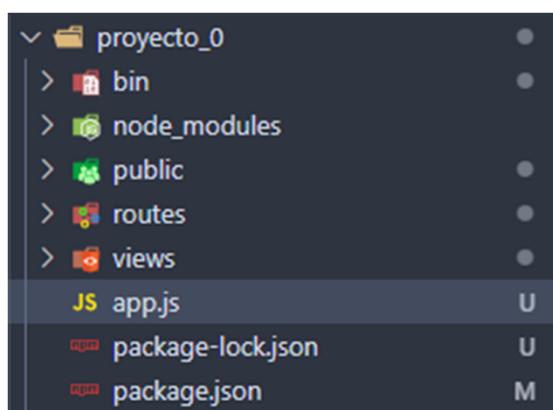
# Modulo {express-generator}

- Para crear rápidamente un esqueleto de aplicación.

```
[kralos]--[y main ≡ ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\projeto_0]
o ▶ npx express-generator

warning: the default view engine will not be jade in future releases
warning: use `--view=jade` or `--help` for additional options

destination is not empty, continue? [y/N] █
```



```
destination is not empty, continue? [y/N] y
●
  create : .
  create : ./package.json
  create : ./app.js
  create : ./public/javascripts
  create : ./public/images
  create : ./public/stylesheets
  create : ./public/stylesheets/style.css
  create : ./public
  create : ./routes
  create : ./routes/index.js
  create : ./routes/users.js
  create : ./views
  create : ./views/index.jade
  create : ./views/layout.jade
  create : ./views/error.jade
  create : ./bin
  create : ./bin/www

  install dependencies:
    > cd . && npm install

  run the app:
    > SET DEBUG=projeto-0:* & npm start
```

# Modulo {express-generator}

- **npx** es también una herramienta CLI cuyo propósito es facilitar la instalación y la gestión de las dependencias alojadas en el registro npm.
- Por ejemplo, tenemos en nuestro proyecto “ESLint” antes teníamos que buscar el binario en node\_modules.

```
./node_modules/.bin/eslint yourfile.js
```

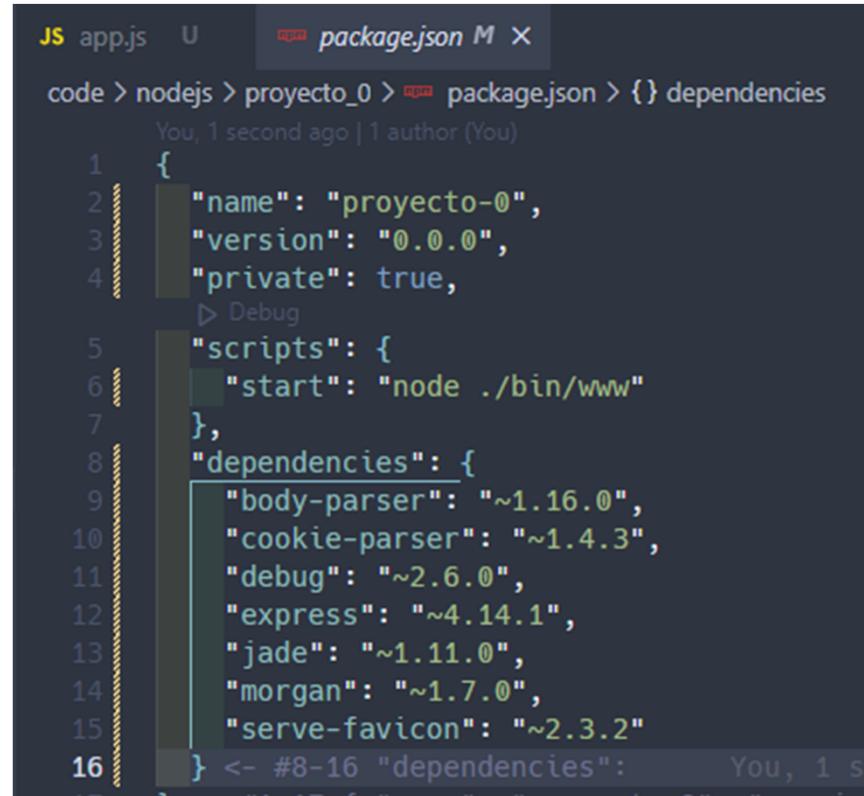
# Modulo {express-generator}

- Ahora npx lo buscará por nosotros.

```
npx eslint yourfile.js
```

- En nuestro proyecto instala y ejecuta express-generator, y como podemos ver reemplaza y agrega código.

# Modulo {express-generator}



The screenshot shows a terminal window with the following content:

```
JS app.js U package.json M X
code > nodejs > proyecto_0 > package.json > {} dependencies
    You, 1 second ago | 1 author (You)
1  {
2      "name": "proyecto-0",
3      "version": "0.0.0",
4      "private": true,
5      "scripts": {
6          "start": "node ./bin/www"
7      },
8      "dependencies": {
9          "body-parser": "~1.16.0",
10         "cookie-parser": "~1.4.3",
11         "debug": "~2.6.0",
12         "express": "~4.14.1",
13         "jade": "~1.11.0",
14         "morgan": "~1.7.0",
15         "serve-favicon": "~2.3.2"
16     } <- #8-16 "dependencies": You, 1 s
```

# Modulo {express-generator}

```
[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\projeto_0]
● ▶ npm install
npm WARN deprecated transformers@2.1.0: Deprecated, use jstransformer
npm WARN deprecated constantinople@3.0.2: Please update to at least constantinople 3.1.1
npm WARN deprecated jade@1.11.0: Jade has been renamed to pug, please install the latest version of pug instead of jade
○ added 61 packages, removed 14 packages, changed 26 packages, and audited 112 packages in 9s
  19 vulnerabilities (1 low, 1 moderate, 11 high, 6 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Luego de muchos “npm audit fix –force”

```
● ▶ npm audit fix --force
npm WARN using --force Recommended protections disabled.

up to date, audited 86 packages in 1s

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

  found 0 vulnerabilities
[kralos]--[¶]main = ● ]
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\projeto_0]
○ ▶ ||
```

# Modulo {express-generator}

- Ejecutando la App

- CMD

```
> set DEBUG=app:* & npm start
```

- PowerShell

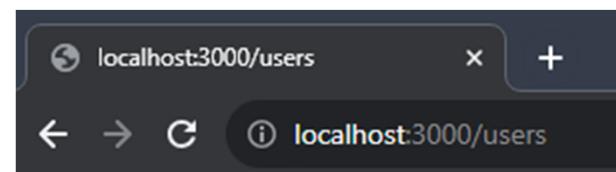
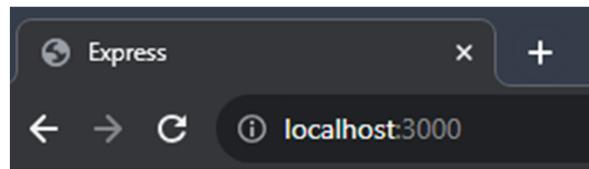
```
PS> $env:DEBUG='myapp:*'; npm start
```

# Modulo {express-generator}

```
[kralos]--[~] main = ●
[D:\umar\Asignaturas\Semestre 23-24 A\912_TI_1\Notas_TI1\code\nodejs\proyecto_0]
▶ $env:DEBUG='app:*'; npm start

> proyecto-0@0.0.0 start
> node ./bin/www

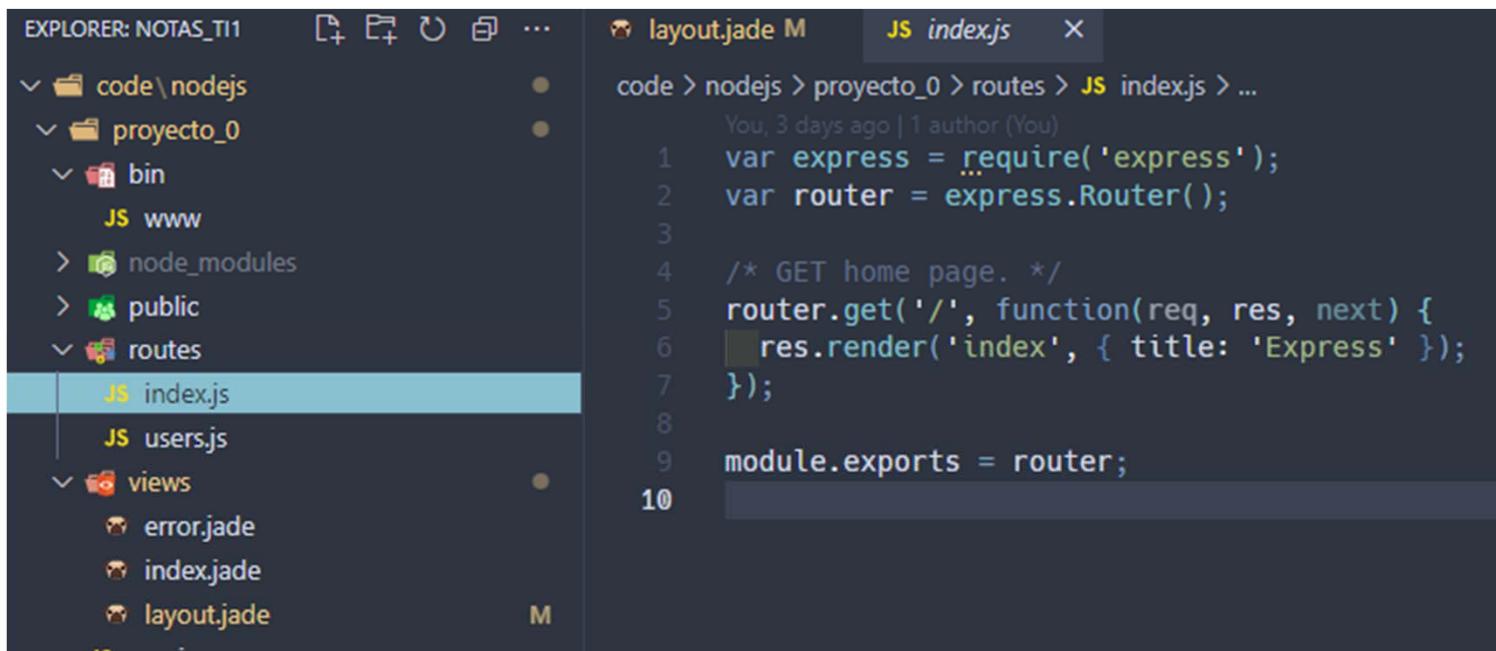
GET / 200 57.282 ms - 170
GET /stylesheets/style.css 200 29.348 ms - 111
□
```



**Express**

Welcome to Express

# Parametros 'title'



The screenshot shows the VS Code interface with the Explorer, Editor, and Terminal panes.

**EXPLORER:** NOTAS\_TI1

- code\nodejs
- proyecto\_0
  - bin
  - www
  - node\_modules
  - public
  - routes
    - index.js
    - users.js
  - views
    - error.jade
    - index.jade
    - layout.jade

**EDITOR:** layout.jade M    JS index.js X

code > nodejs > proyecto\_0 > routes > JS index.js > ...  
You, 3 days ago | 1 author (You)

```
1 var express = require('express');
2 var router = express.Router();
3
4 /* GET home page. */
5 router.get('/', function(req, res, next) {
6   res.render('index', { title: 'Express' });
7 });
8
9 module.exports = router;
10
```

# Un poco de jade

The image shows a code editor interface with a sidebar and two main panes. The sidebar on the left lists project files: code\nodejs, proyecto\_0 (with subfolders bin, www, node\_modules, public, routes containing index.js and users.js), views (with error.jade, index.jade), and layout.jade. The main pane on the left displays the content of layout.jade, which includes doctype html, head (title= title, link rel='stylesheet' href='/stylesheets/style.css'), body (nav (ul (li a href="/" href="#">Inicio), li a href="/contacto.html" href="#">Contacto)), and a block content section. The main pane on the right shows a browser window titled 'Express' at localhost:3000. The page content includes a navigation menu with links to 'Inicio' and 'Contacto', followed by the text 'Welcome to Express'.

```
EXPLORER: NOTAS_TI1      🔍 ⌂ ⌂ ⌂ ⌂ ... 🏷️ layout.jade M ✎ JS index.js
code > nodejs > proyecto_0 > views > layout.jade
You, 1 minute ago | 1 author (You)
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    nav
      ul
        li
          a(href="/") Inicio
        li
          a(href="/contacto.html") Contacto
    block content

code > nodejs > proyecto_0 > views > layout.jade
You, 1 minute ago | 1 author (You)
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    nav
      ul
        li
          a(href="/") Inicio
        li
          a(href="/contacto.html") Contacto
    block content

Express
localhost:3000
• Inicio
• Contacto
Welcome to Express
```

# Un poco de jade

## Basics

edit jade ↗

```
doctype html
html
  head
    title my jade template
  body
    h1 Hello #{name}
```

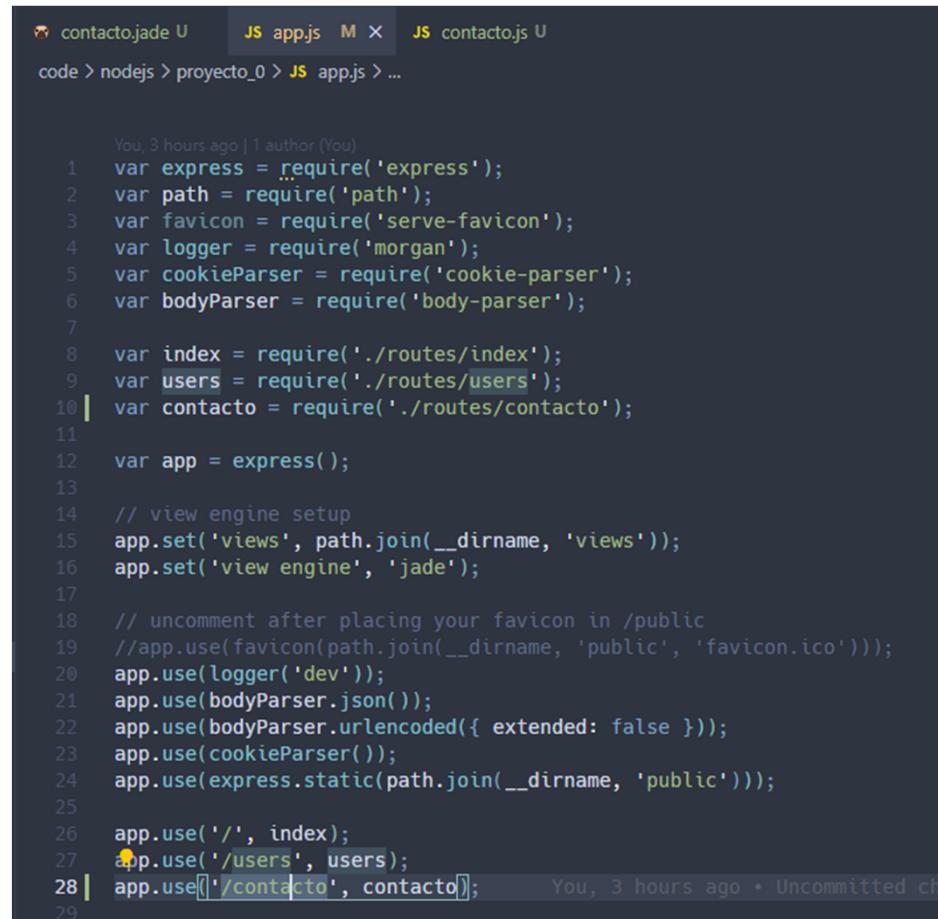
```
{"name": "Bob"}
```

edit data ↗

↙ html output

```
<!DOCTYPE html>
<html>
  <head>
    <title>my jade template</title>
  </head>
  <body>
    <h1>Hello Bob</h1>
  </body>
</html>
```

# Un poco de jade

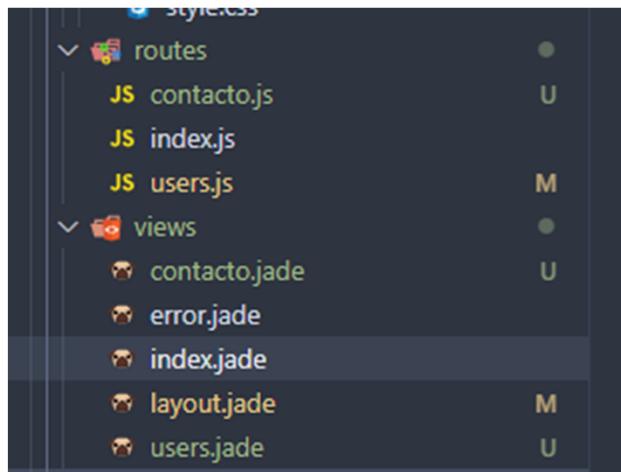


The screenshot shows a terminal window with the following details:

- File tabs: contacto.jade U, JS app.js M X, JS contacto.js U.
- Path: code > nodejs > proyecto\_0 > JS app.js > ...
- Author: You, 3 hours ago | 1 author (You)
- Code content (lines 1-29):

```
You, 3 hours ago | 1 author (You)
1 var express = require('express');
2 var path = require('path');
3 var favicon = require('serve-favicon');
4 var logger = require('morgan');
5 var cookieParser = require('cookie-parser');
6 var bodyParser = require('body-parser');
7
8 var index = require('./routes/index');
9 var users = require('./routes/users');
10 var contacto = require('./routes/contacto');
11
12 var app = express();
13
14 // view engine setup
15 app.set('views', path.join(__dirname, 'views'));
16 app.set('view engine', 'jade');
17
18 // uncomment after placing your favicon in /public
19 //app.use(favicon(path.join(__dirname, 'public', 'favicon.ico')));
20 app.use(logger('dev'));
21 app.use(bodyParser.json());
22 app.use(bodyParser.urlencoded({ extended: false }));
23 app.use(cookieParser());
24 app.use(express.static(path.join(__dirname, 'public')));
25
26 app.use('/', index);
27 app.use('/users', users);
28 app.use(['/contacto', contacto]);      You, 3 hours ago • Uncommitted ch
29
```

# Un poco de jade



contacto.jade U JS app.js M JS contacto.js U X

```
code > nodejs > proyecto_0 > routes > JS contacto.js > ...
1 var express = require('express');
2 var router = express.Router();
3
4 /* GET users listing. */
5 router.get('/', function(req, res, next) {
6   res.render('contacto', { title: 'Express Contacto' });
7 });
8
9 module.exports = router;
```

contacto.jade U X JS app.js M JS contacto.js U

```
code > nodejs > proyecto_0 > views > contacto.jade
1 extends layout
2
3 block content
4   h1 Hola
5
```

# Tips {package.json}

- "type": "module"

```
import express from 'express';
```

- "type": "commonjs"

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

# API con Express

```
JS index.js U X
code > nodejs > proyecto_1 > JS index.js > ...
1  const express = require('express');
2  const app = express();
3  const hostname = '127.0.0.1';
4  const port = 3000;
5  app.use(express.json());
6
7  const students = [
8      {id: 1, name: "Juan", age: 20, enroll: true},
9      {id: 2, name: "Pedro", age: 21, enroll: false},
10     {id: 3, name: "María", age: 23, enroll: false},
11 ];
12
13 app.get('/', (req, res) => {
14     res.send('Hello API Node js');
15 });
16
17 app.get('/api/students', (req, res) => {
18     res.send(students);
19 });
20 app.get('/api/students/:id', (req, res) => {
21     const student = students.find(c => c.id === parseInt(req.params.id));
22     if (!student) return res.status(404).send('Estudiante no encontrado');
23     else res.send(student);
24 });
25 app.post('/api/students/add/', (req, res) => {
26     const student = {
27         id: students.length + 1,
28         name: req.body.name,
29         age: parseInt(req.body.age),
30         enroll: (req.body.enroll === 'true'),
31     };
32     students.push(student);
33     res.send(student);
34 });
35 app.delete('/api/students/:id', (req, res) => {
36     const student = students.find(c => c.id === parseInt(req.params.id));
37     if (!student) return res.status(404).send('Estudiante no encontrado');
38     const index = students.indexOf(student);
39     students.splice(index, 1);
40     res.send(student);
41 });
42
43 app.listen(port, hostname, () => {
44     console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
45 });


```

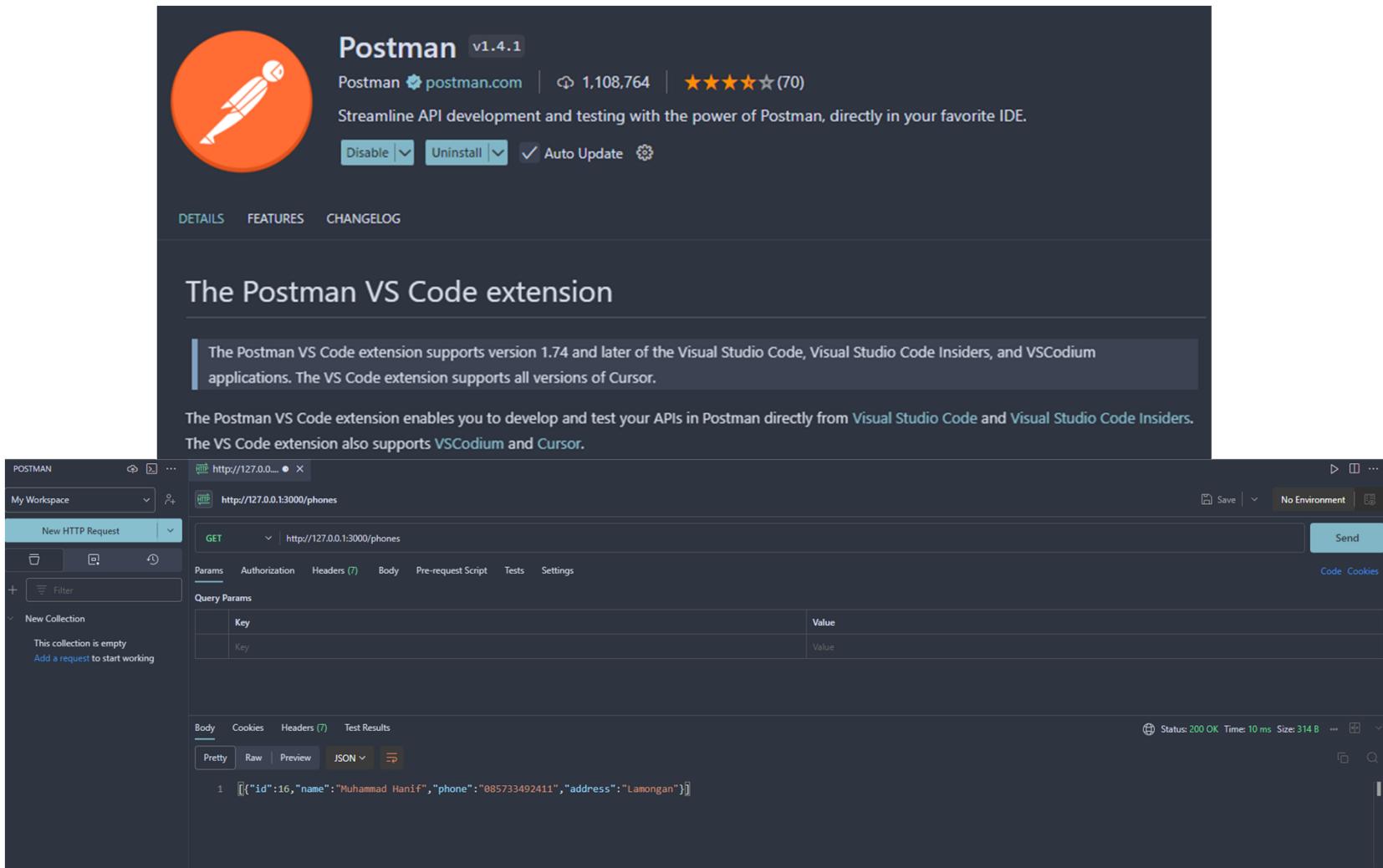
Listar

Buscar

Insertar

Eliminar

# Postman on VSCode



The screenshot shows the Postman VS Code extension interface. At the top, there's a header with the Postman logo (orange circle with a white rocket), the version v1.4.1, download stats (1,108,764), and a 4-star rating (70). Below the header are buttons for 'Disable' and 'Uninstall' and a checked 'Auto Update' option. A progress bar indicates the update is 100% complete.

Below the header, there are three tabs: 'DETAILS', 'FEATURES', and 'CHANGELOG'. The 'FEATURES' tab is currently selected, displaying the following text:

**The Postman VS Code extension**

The Postman VS Code extension supports version 1.74 and later of the Visual Studio Code, Visual Studio Code Insiders, and VSCodium applications. The VS Code extension supports all versions of Cursor.

The Postman VS Code extension enables you to develop and test your APIs in Postman directly from Visual Studio Code and Visual Studio Code Insiders. The VS Code extension also supports VSCodium and Cursor.

The main workspace shows a request list with one item: 'GET http://127.0.0.1:3000/phones'. The 'Body' tab is selected, showing a JSON response:

```
1 [{"id":16,"name":"Muhammad Hanif","phone":"085733492411","address":"Lamongan"}]
```

The status bar at the bottom right shows: Status: 200 OK Time: 10 ms Size: 314 B.

# Postman on VSCode

The image displays two screenshots of the Postman extension for VSCode. Both screenshots show a dark-themed interface with a sidebar on the left containing collections and requests.

**Screenshot 1 (Top): POST Request**

- Request Details:** POST to `http://127.0.0.1:3000/api/v1/add/phone`
- Body:** Raw JSON (selected) with the following content:

```
1 [ {  
2   "name": "Pedro",  
3   "phone": "6666-666-666",  
4   "address": "all"  
5 } ]
```
- Response:** Status: 200 OK, Time: 52 ms, Size: 264 B. Message: "Insert operation successful !!!" with a thumbs-up emoji.

**Screenshot 2 (Bottom): GET Request**

- Request Details:** GET to `http://127.0.0.1:3000/api/v1/phones`
- Query Params:** A table with one row and two columns: "Key" (Value) and "Value" (Value).
- Response:** Status: 200 OK, Time: 14 ms, Size: 499 B. Response body (JSON):

```
1 [ {  
2   "id": 4, "name": "Juan", "phone": "9999999999", "address": "s/n"}, {  
3   "id": 16, "name": "Muhammad Hanif", "phone": "085733492411", "address": "Lamongan"}, {  
4   "id": 21, "name": "Juan", "phone": "9999999999",  
5   "address": "s/n"}, {  
6   "id": 22, "name": "Pedro", "phone": "6666-666-666", "address": "all"} ]
```

# API con Express

- Trabajo 2do Parcial
  - Aplicar buenas prácticas a la API de Express
  - MariaDB Server o MySQL
- <https://vicente-aguilera-perez.medium.com/las-10-mejores-practicas-para-nombrar-api-rest-endpoints-48b8bcbd1397>
- <https://mascandobits.es/programacion/buenas-practicas-para-el-buen-diseno-de-una-api-restful/>

```

code > nodejs > proyecto_2 > package.json > ...
1  {
2    "name": "proyecto_2",
3    "version": "1.0.0",
4    "main": "index.js", →
5      ▷ Debug
6    "scripts": {
7      "start": "nodemon index.js",
8      "test": "echo \"Error: no test specified\" && exit 1"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "description": "",
14   "dependencies": {
15     "express": "^4.21.1"
16   },
17   "devDependencies": {
18     "nodemon": "^3.1.7"
19   }
}

```

code > nodejs > proyecto\_2 > js index.js > ...

```

1  const express = require('express');
2  const app = express();
3  const hostname = '127.0.0.1';
4  const port = 3000;
5
6  app.get('/', (req, res) => {
7    res.send('Hello World');
8  })
9
10 app.listen(port, hostname, () => {
11   console.log(`El servidor se está ejecutando en http://${hostname}:${port}/`);
12 })

```

○ [kralos]--[✖ main = ● ]  
[D:\umar\Asignaturas\Semestre 24-25 A\712\_TW\_II\Notas\_TW2\code\nodejs\proyecto\_2]

▶ npm run start

> proyecto\_2@1.0.0 start  
> nodemon 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 index.js`  
El servidor se está ejecutando en http://127.0.0.1:3000/  
[nodemon] restarting due to changes...  
[nodemon] starting `node index.js`  
El servidor se está ejecutando en http://127.0.0.1:3000/

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CONSOLE

● [kralos]--[✖ main = ● ]  
[D:\umar\Asignaturas\Semestre 24-25 A\712\_TW\_II\Notas\_TW2\code\nodejs\proyecto\_2]

▶ npm install express

added 65 packages, and audited 66 packages in 5s

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

found 0 vulnerabilities

● [kralos]--[✖ main = ● ]  
[D:\umar\Asignaturas\Semestre 24-25 A\712\_TW\_II\Notas\_TW2\code\nodejs\proyecto\_2]

▶ npm install nodemon --save-dev

added 29 packages, and audited 95 packages in 6s

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

found 0 vulnerabilities

# Referencias

- *Manual de NodeJS.* (n.d.). Desarrolloweb.com. Retrieved October 9, 2023, from <https://desarrolloweb.com/manuales/manual-nodejs.html>
- *Nodejs — Chuletas.* (n.d.). Github.io. Retrieved October 10, 2023, from <https://jolav.github.io/chuletas/nodejs/>
- <https://naltatis.github.io/jade-syntax-docs/>