Servidor node.js

```
const http = require('http')

const server = http.createServer((req, res)=>{
    res.writeHead(200)
    res.write('<h1>Respuesta de mi servidor</h1>')
    res.end()
})

server.listen(3000)
console.log('Respuesta del server')
```

1

Crea un servidor

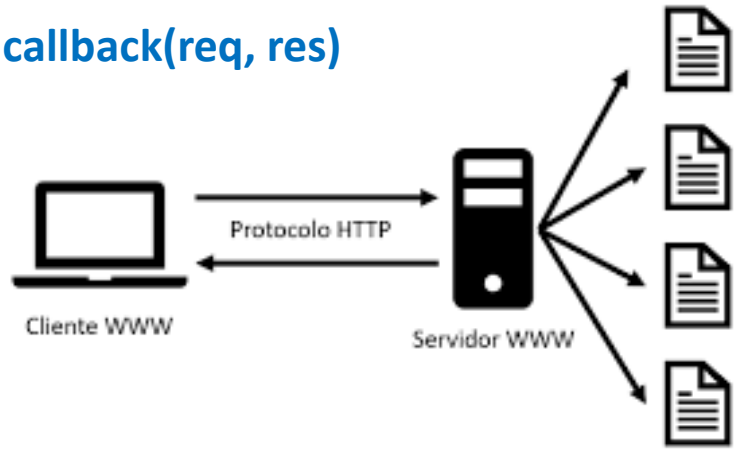
2

Escucha en el puerto **3000**

3

Cada vez que recibas un **pedido http** ejecuta el **callback(req, res)**

...



Paquetes externos

- **npm** es el **sistema** de gestión de paquetes por defecto para Node.js
- Cuando nos referimos a “paquetes” en realidad a lo que se hace referencia, es a paquetes de código que vivirán en una carpeta llamada **“node_modules”** y tendrán una configuración que los llamará dentro de un archivo llamado **“package.json”**.

Se ejecuta al crear la carpeta del proyecto

- El autor
- El título del proyecto
- La licencia
- El repositorio
- Las dependencia
- Scripts
- Test
- Entre otros

Paquetes externos

- **Por ejemplo:**

```
Ctrl+ C (detener servidor)

 > npm install nodemon -d
```

Permite reiniciar el servidor automáticamente

Solo disponible en ambientes de desarrollo

- *package.json: (archivo de configuración)*

```
"scripts": {
    "start": "nodemon index.js"
  },
```

- *> npm start*

```
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
```

https://www.npmjs.com/

A tener en cuenta 

- *Cuando desplegamos nuestro servidor en un servicio de la nube,* ***Heroku*** *por ejemplo, dejamos solo la carpeta* ***src y*** *los archivos****: package.json y package-lock.json***

- *Luego desde el servicio en la nube, ejecutamos:*

  - ***npm install***

- *Se descargan automáticamente todos los paquetes externos configurados como dependencias de producción, no de desarrollo*

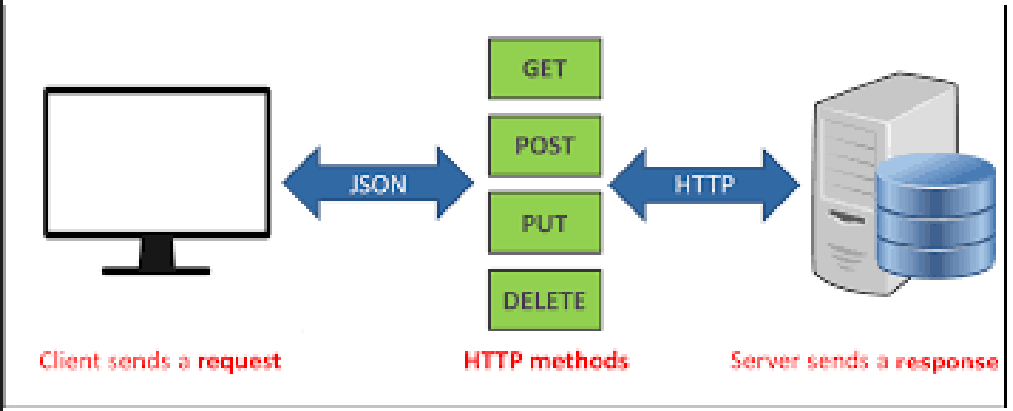
A programar…

Vamos a practicar

Caso práctico integrador A

- Desarrollar una API Rest con Node.js que permita gestionar posts según las siguientes entradas:

  - **GET: /** → Muestra solo un mensaje con el nombre del servidor: **Práctico POSTS**
  - **GET /posts** → Retorna el listado de posts cargados hasta el momento
  - **POST /posts** → Crear un post y devolverlo como respuesta de la petición



Usar POSTMAN para hacer pruebas

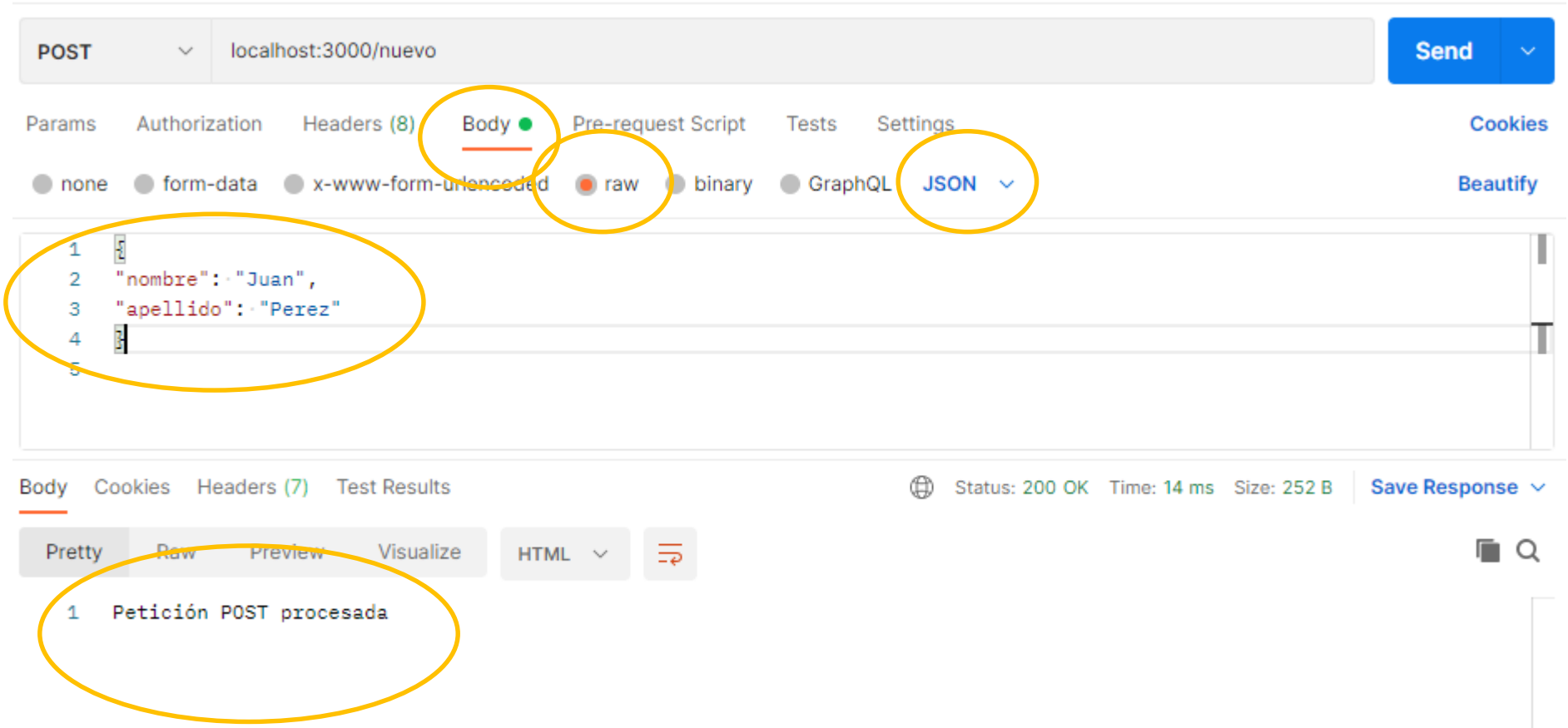
- **POSTMAN** es una herramienta que permite principalmente crear peticiones sobre APIs de una forma muy sencilla y poder, de esta manera, probar las APIs.

https://www.postman.com/downloads/.

- **Crear Peticiones** de forma gráfica
- Permite definir **Colecciones**
- **Gestiona la Documentación**
- **Entorno Colaborativo**
- **Genera código de invocación**
- **Establecer variables**
- **Soporta Ciclo Vida API**
- **Crear mockups**

POSTMAN

Enviar y recibir datos JSON



A programar…

Vamos a practicar

Caso práctico integrador B

- Refactorizar la API Rest del **caso práctico integrador A** de tal manera que los post se persistan en la base de datos **posts.db** utilizada en los prácticos anteriores. Completar las funcionalidades necesarias para
  - **GET: /** → Muestra solo un mensaje con el nombre del servidor: **Práctico POSTS**
  - **GET /posts** → Retorna el listado de posts cargados hasta el momento
  - **POST /posts** → Crear un post y devolverlo como respuesta de la petición
  - **PUT /posts** → Actualizar un post por Id
  - **DELETE /posts** → Eliminar un post por Id