

Ayudantía 3

NPM, Node.js y Express.js



Sebastián Riquelme

¿Qué es Node.js?

Javascript en el backend. En nuestro caso, se ejecuta en el servidor.


```
1 // Promise from setTimeout
2 const afterSomeTime = (time) => new Promise(resolve => {
3   setTimeout(() => {
4     resolve(true);
5   }, time);
6 });
7 const callAfterSomeTime = (callback, time) => afterSomeTime(time).then(callback);
8
9 callAfterSomeTime(() => console.log('Hello after 1500ms', 1500));
10
11 const getData = async (url) => fetch(url);
12
13 document
14   .querySelector('#submit')
15   .addEventListener('click', function() {
16     const name = document.querySelector('#name').value;
17
18     // send to backend
19     const user = await fetch(`/users?name=${name}`);
20     const posts = await fetch(`/posts?userId=${user.id}`);
21     const comments = await fetch(`/comments?post=${posts[0].id}`);
22     //display comments on DOM
```



Express.js

Express.js es el framework backend más popular para Node.js.

- Render HTML
- API
- Enrutamiento



```
1  const express = require("express");
2  const app = express();
3  app.post('/hola', function (req, res) {
4    res.send('[POST]Saludos desde express');
5  });
6  app.get('/hola', function (req, res) {
7    res.send('[GET]Saludos desde express');
8  });
9  app.listen(3000, () => {
10   console.log("El servidor está inicializado en el puerto 3000");
11 });
12
```

Instalar NPM

Mediante gestor de versiones de node NVM. Fuente: [Guia instalación](#)

Para descargarlo (Puedes ver la ultima version del curl en <https://github.com/nvm-sh/nvm>):
`curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash`

Para instalarlo:
`source ~/.bashrc`

Para ver las versiones disponibles de node:
`nvm list-remote`

Instalar NPM

Instalar la última LTS (soporte a largo plazo en inglés, Long Term Support)

`npm install -g nvm`
`nvm install v16.17.1`

```
ubuntu@ayudantia: ~  
-----  
v16.10.0  
v16.11.0  
v16.11.1  
v16.12.0  
v16.13.0 (LTS: Gallium)  
v16.13.1 (LTS: Gallium)  
v16.13.2 (LTS: Gallium)  
v16.14.0 (LTS: Gallium)  
v16.14.1 (LTS: Gallium)  
v16.14.2 (LTS: Gallium)  
v16.15.0 (LTS: Gallium)  
v16.15.1 (LTS: Gallium)  
v16.16.0 (LTS: Gallium)  
v16.17.0 (LTS: Gallium)  
v16.17.1 (Latest LTS: Gallium)  
v17.0.0  
v17.0.1  
v17.1.0  
v17.2.0  
v17.3.0  
v17.3.1  
v17.4.0  
v17.5.0  
v17.6.0
```

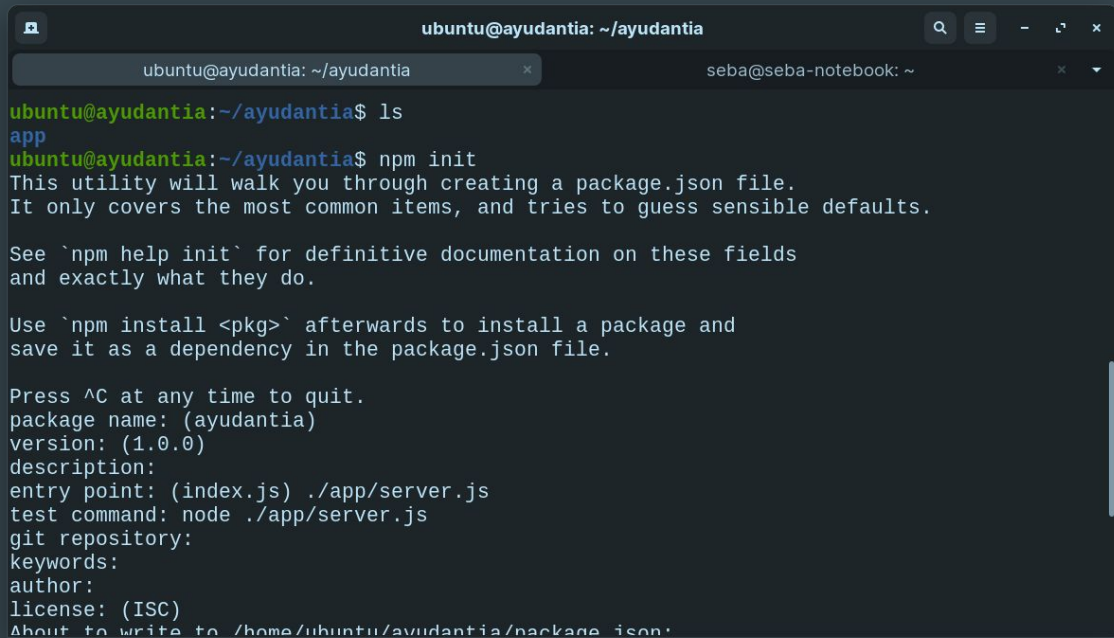
Instalar NPM

Para ver la versión usar:
node -v

```
ubuntu@ayudantia: ~  
v18.9.1  
ubuntu@ayudantia:~$ nvm install v16.17.1  
Downloading and installing node v16.17.1...  
Downloading https://nodejs.org/dist/v16.17.1/node-v16.17.1-linux-x64.tar.xz...  
##### 100.0%  
Computing checksum with sha256sum  
Checksums matched!  
Now using node v16.17.1 (npm v8.15.0)  
Creating default alias: default -> v16.17.1  
ubuntu@ayudantia:~$ nvm list  
->      v16.17.1  
default -> v16.17.1  
iojs -> N/A (default)  
unstable -> N/A (default)  
node -> stable (-> v16.17.1) (default)  
stable -> 16.17 (-> v16.17.1) (default)  
lts/* -> lts/gallium (-> v16.17.1)  
lts/argon -> v4.9.1 (-> N/A)  
lts/boron -> v6.17.1 (-> N/A)  
lts/carbon -> v8.17.0 (-> N/A)  
lts/dubnium -> v10.24.1 (-> N/A)  
lts/erbium -> v12.22.12 (-> N/A)  
lts/fermium -> v14.20.1 (-> N/A)  
lts/gallium -> v16.17.1  
ubuntu@ayudantia:~$ node -v  
v16.17.1  
ubuntu@ayudantia:~$ |
```

Iniciar nuestro proyecto

npm init



```
ubuntu@ayudantia: ~/ayudantia
ubuntu@ayudantia: ~/ayudantia$ ls
app
ubuntu@ayudantia:~/ayudantia$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

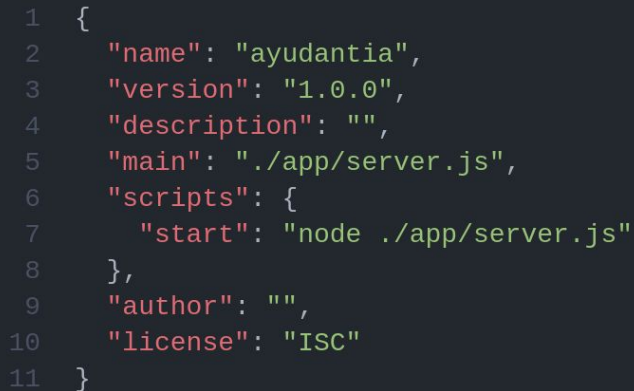
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (ayudantia)
version: (1.0.0)
description:
entry point: (index.js) ./app/server.js
test command: node ./app/server.js
git repository:
keywords:
author:
license: (ISC)
About to write to /home/ubuntu/ayudantia/package.json:
```

Archivo package.json

Package.json guardará las dependencias de nuestro proyecto. Si necesitamos instalar nuestro proyecto en otra máquina, podemos usar el comando `npm -i`.

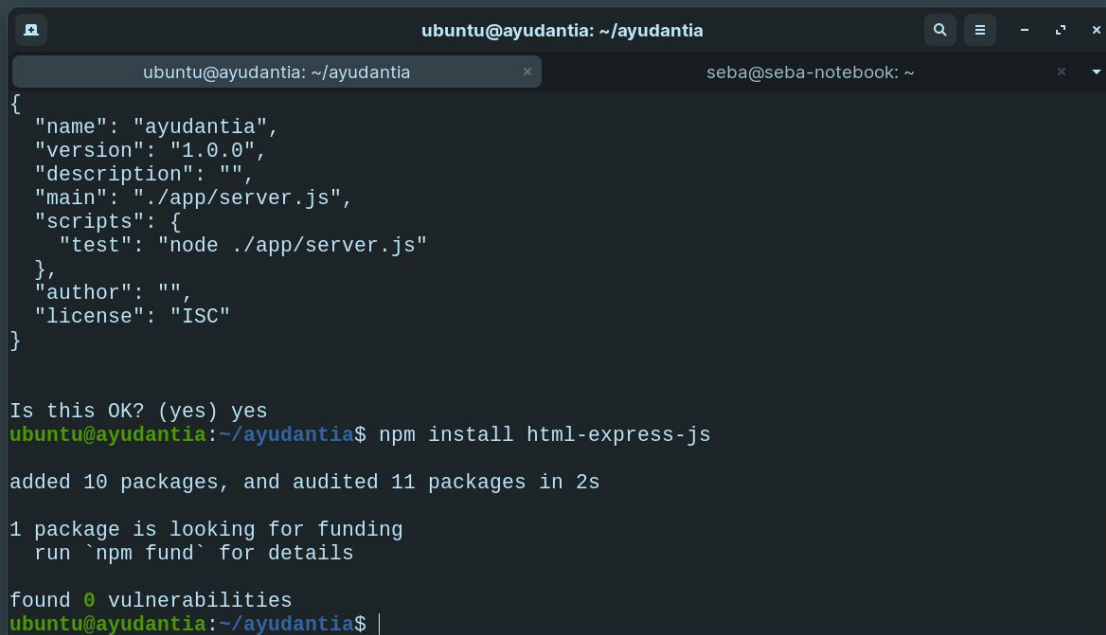
Archivo package.json antes de
instalar cualquier dependencia



```
1 {  
2   "name": "ayudantia",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "./app/server.js",  
6   "scripts": {  
7     "start": "node ./app/server.js"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }
```


Instalar html-express-js

npm install html-express-js



A terminal window titled 'ubuntu@ayudantia: ~/ayudantia' showing the installation of 'html-express-js'. The window has a dark theme and a search bar in the top right. The terminal output shows a JSON package configuration, a confirmation prompt, the installation command, and the results of the installation.

```
ubuntu@ayudantia: ~/ayudantia
{
  "name": "ayudantia",
  "version": "1.0.0",
  "description": "",
  "main": "./app/server.js",
  "scripts": {
    "test": "node ./app/server.js"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
ubuntu@ayudantia:~/ayudantia$ npm install html-express-js

added 10 packages, and audited 11 packages in 2s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ubuntu@ayudantia:~/ayudantia$ |
```

Instalar dependencias

`npm install <nombre del paquete>`

Se agrega el apartado `dependencies` a nuestro archivo `package.json`, así, con `npm install` podemos instalar la dependencia que queramos y se agregará a nuestro `package.js`

```
1  {  
2    "name": "ayudantia",  
3    "version": "1.0.0",  
4    "description": "",  
5    "main": "./app/server.js",  
6    "scripts": {  
7      "start": "node ./app/server.js"  
8    },  
9    "author": "",  
10   "license": "ISC",  
11   "dependencies": {  
12     "html-express-js": "^1.1.1"  
13   }  
14 }  
15
```

Trabajando con render de html desde express

Para ello haremos uso del siguiente proyecto <https://github.com/markcellus/html-express-js>

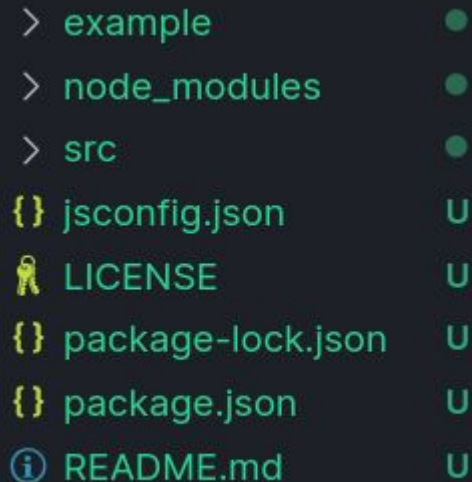
Podemos descargar el proyecto con el siguiente comando:

git clone <https://github.com/markcellus/html-express-js> .

Luego para instalar las dependencias:

npm i

Deberíamos tener estas carpetas:



A screenshot of a file explorer window with a dark theme. It shows a list of files and folders. The first three items are folders: 'example', 'node_modules', and 'src', each preceded by a green right-pointing chevron and followed by a green circle icon. The remaining items are files: 'jsconfig.json' (preceded by a green curly brace icon), 'LICENSE' (preceded by a yellow key icon), 'package-lock.json' (preceded by a green curly brace icon), 'package.json' (preceded by a green curly brace icon), and 'README.md' (preceded by a blue circle icon with an 'i'). Each file is followed by a green 'U' icon.

- > example
- > node_modules
- > src
- { } jsconfig.json
- 🔑 LICENSE
- { } package-lock.json
- { } package.json
- ⓘ README.md

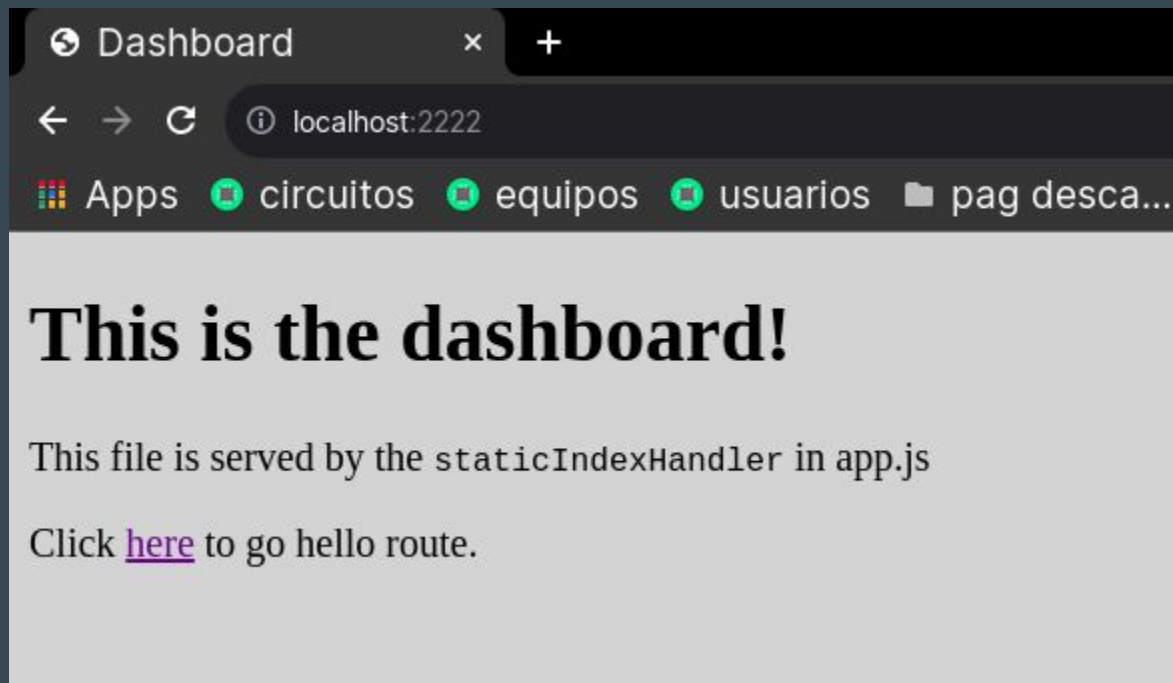
Ejecutamos el proyecto para ver que todo esté bien

```
seba@seba-notebook: ~/Escritorio/Ayudantía/ayudantia 5
seba@seba-notebook:~/Escritorio/Ayudantía/ayudantia 5$ npm start

> html-express-js@1.1.1 start
> node ./example/server.js

constante de directorio __dirname: /home/seba/Escritorio/Ayudantía/ayudantia 5
Starting WebSocket Server
Server started at http://localhost:2222
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Sending message to 0 connection(s): reload
Reload client connected to server
Reload client connected to server
```

Visitamos ip:2222



Server.js

Es el servidor de nuestra app y ejecuta nuestra app en la dirección que se indica.



```
1 import app from './app.js';
2 import reload from 'reload';
3 import chokidar from 'chokidar';
4
5 const port = 2222;
6
7 // reload browser on file changes
8 // Ejecuta nuestra app y la recarga ante cualquier cambio en un archivo
9 reload(app, { verbose: true })
10 .then(function (reloadReturned) {
11   chokidar.watch(['./src', './example']).on('all', (*event, path*) => {
12     reloadReturned.reload();
13   });
14
15   //Inicia nuestra app en la direccion indicada
16   app.listen(port, function () {
17     console.log(`Server started at http://localhost:${port}`);
18   });
19 })
20
21 //catch se ejecuta si ocurre un error
22 .catch(function (err) {
23   console.error('Reload could not start, could not start example app', err);
24 });
```

App.js

Configuramos nuestra app a detalle, con las rutas de nuestras views, los static, etc.

```
1 import express from 'express';
2 import { resolve } from 'path';
3 import htmlExpress, { staticIndexHandler } from '../src/index.js';
4
5 //seteamos el directorio
6 const __dirname = resolve();
7
8 console.log("constante de directorio __dirname: ", __dirname);
9 // home/seba/Escritorio/Ayudantia/ayudantia 5
10
11
12 //Iniciamos la app express
13 const app = express();
14
15 app.engine(
16   'js',
17   htmlExpress({
18     includesDir: 'includes',
19   })
20 );
21
22
23 //view engine js quiere decir que nuestro sistema de vistas será js
24 app.set('view engine', 'js');
```

```
1 // home/seba/Escritorio/Ayudantia/ayudantia 5 = ${__dirname}
2 // Entonces `${__dirname}/example/public`
3 // Es lo mismo que
4 // home/seba/Escritorio/Ayudantia/ayudantia 5/example/public
5
6 // Entonces, indico donde estan mis views
7 app.set('views', `${__dirname}/example/public`);
8
9 // serve all other static files like CSS, images, etc
10 app.use(express.static(`${__dirname}/example/public`));
11
12 // En la ruta /hello renderizo la view hello
13 // con el parametro name con 'word' como contenido, es decir, name: 'word'
14 // visitar localhost/hello
15 app.get('/hello', async function (req, res) {
16   res.render('hello', {
17     name: 'world',
18   });
19 });
20
21 // Automatically serve any index.js file as HTML in the public directory
22 app.use(
23   staticIndexHandler({
24     viewsDir: `${__dirname}/example/public`,
25     notFoundView: 'not-found', // OPTIONAL: defaults to `404/index`
26   })
27 );
28
29 export default app;
30
```

Ejemplo de una view: hello.js

Este es nuestro archivo hello.js

```
1 import { html } from '../../src/index.js';
2
3 export const view = (data, state) => html`
4   <!DOCTYPE html>
5   <html lang="en">
6     <head>
7       ${state.includes.head}
8       <title>Hello!</title>
9     </head>
10
11     <body>
12       <h1>Hello, ${data.name}</h1>
13
14       <p>Click <a href="/">here</a> to go back to the dashboard.</p>
15     </body>
16   </html>
17 `;
18
```

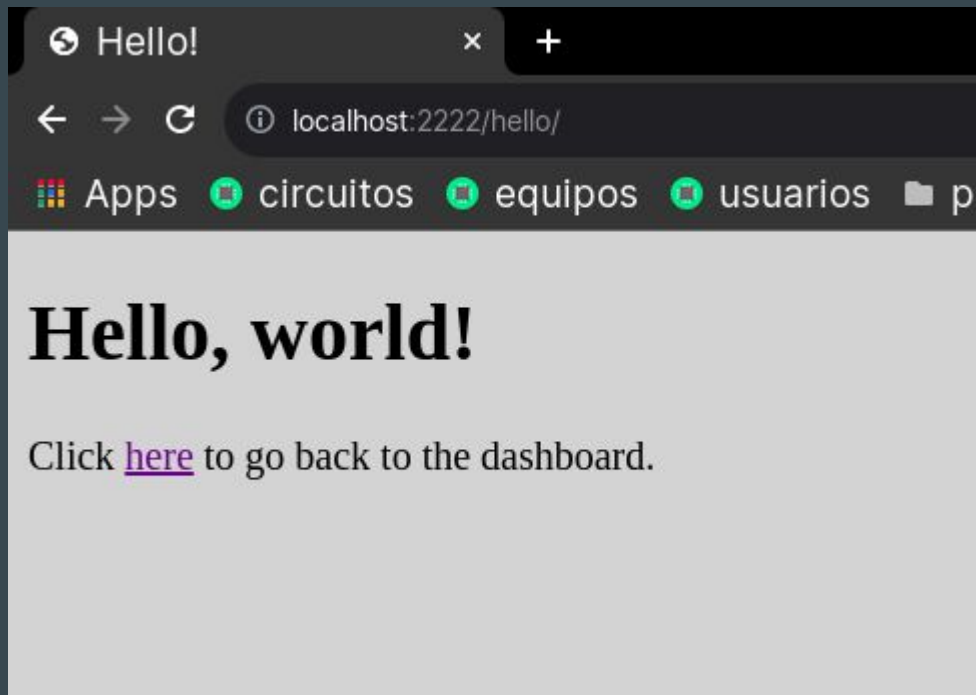

Y así cargamos el hello.js en nuestra app



```
1 // En la ruta /hello renderizo la view hello
2 // con el parámetro name con 'word' como contenido, es decir, name: 'word'
3 // visitar localhost/hello
4 app.get('/hello', async function (req, res) {
5   res.render('hello', {
6     name: 'world',
7   });
8 });
```

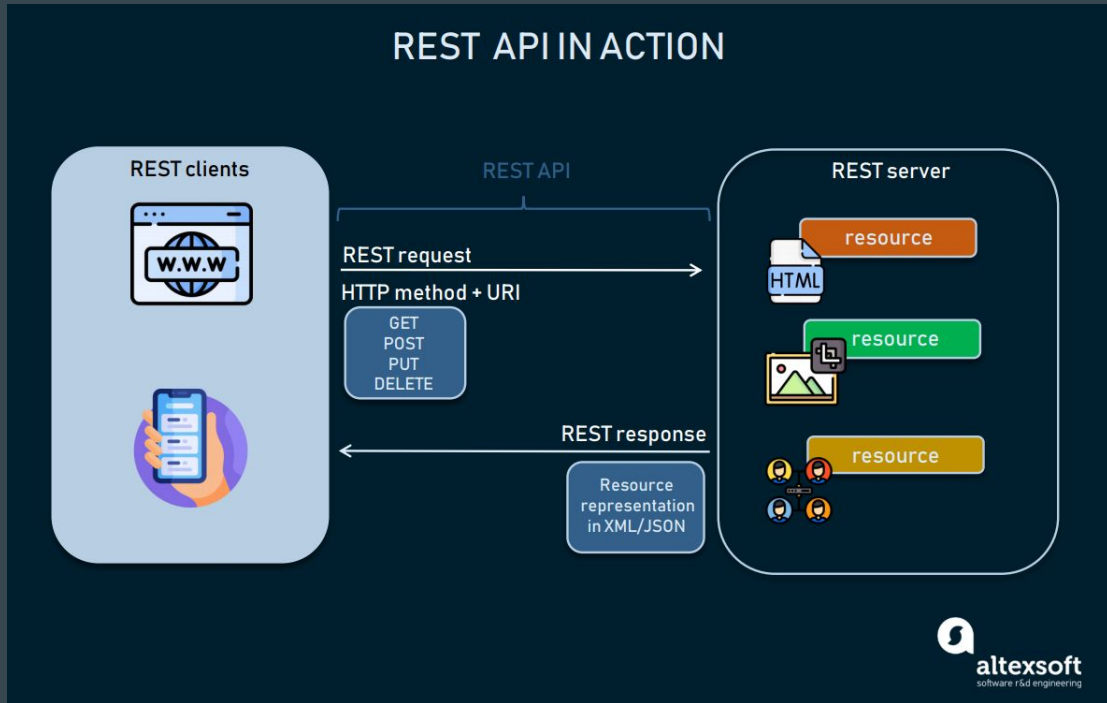
Así se ve la ruta /hello

Noten que la palabra world la pasamos desde nuestra app.js



¿Qué es un API?

- GET = Obtener
- POST = Crear
- PUT = Actualizar
- DELETE = Eliminar



Veamos cómo usarlo con nuestro proyecto

