

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
23     //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
```

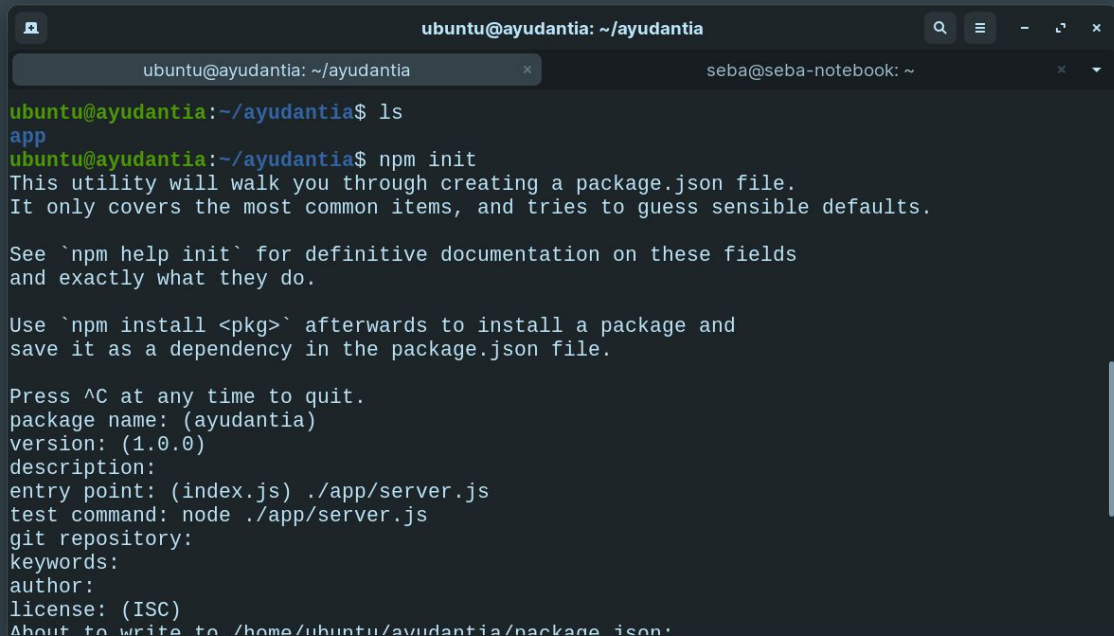
Instalar NPM

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.

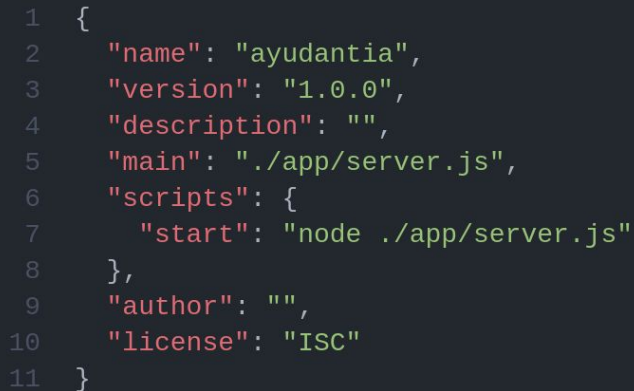
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

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

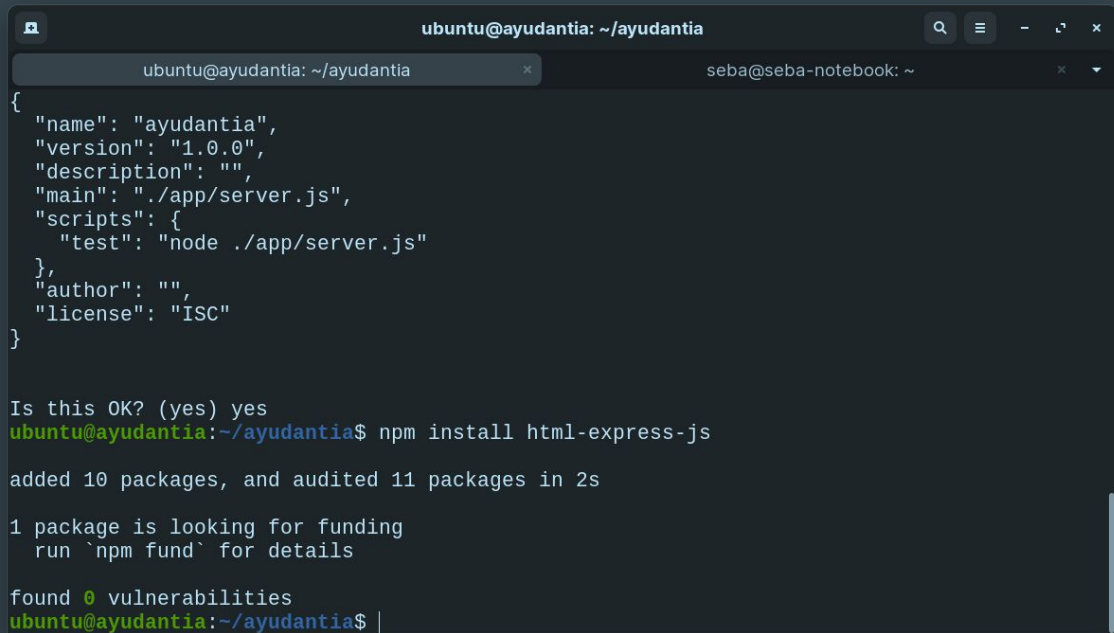


```
1 {  
2   "name": "ayudantia",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "./app/server.js",  
6   "scripts": {  
7     "start": "node ./app/server.js"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }
```


Instalar Express Handlebars

npm i express-handlebars

Link: <https://www.npmjs.com/package/express-handlebars>

A terminal window with a dark background. The title bar shows 'ubuntu@ayudantia: ~/ayudantia'. There are two tabs: 'ubuntu@ayudantia: ~/ayudantia' and 'seba@seba-notebook: ~'. The terminal content shows a JSON object for package.json, a confirmation prompt 'Is this OK? (yes) yes', the command 'npm install html-express-js', and the output 'added 10 packages, and audited 11 packages in 2s'. It also shows '1 package is looking for funding' and 'found 0 vulnerabilities'. The prompt 'ubuntu@ayudantia:~/ayudantia\$' is at the bottom.

```
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`

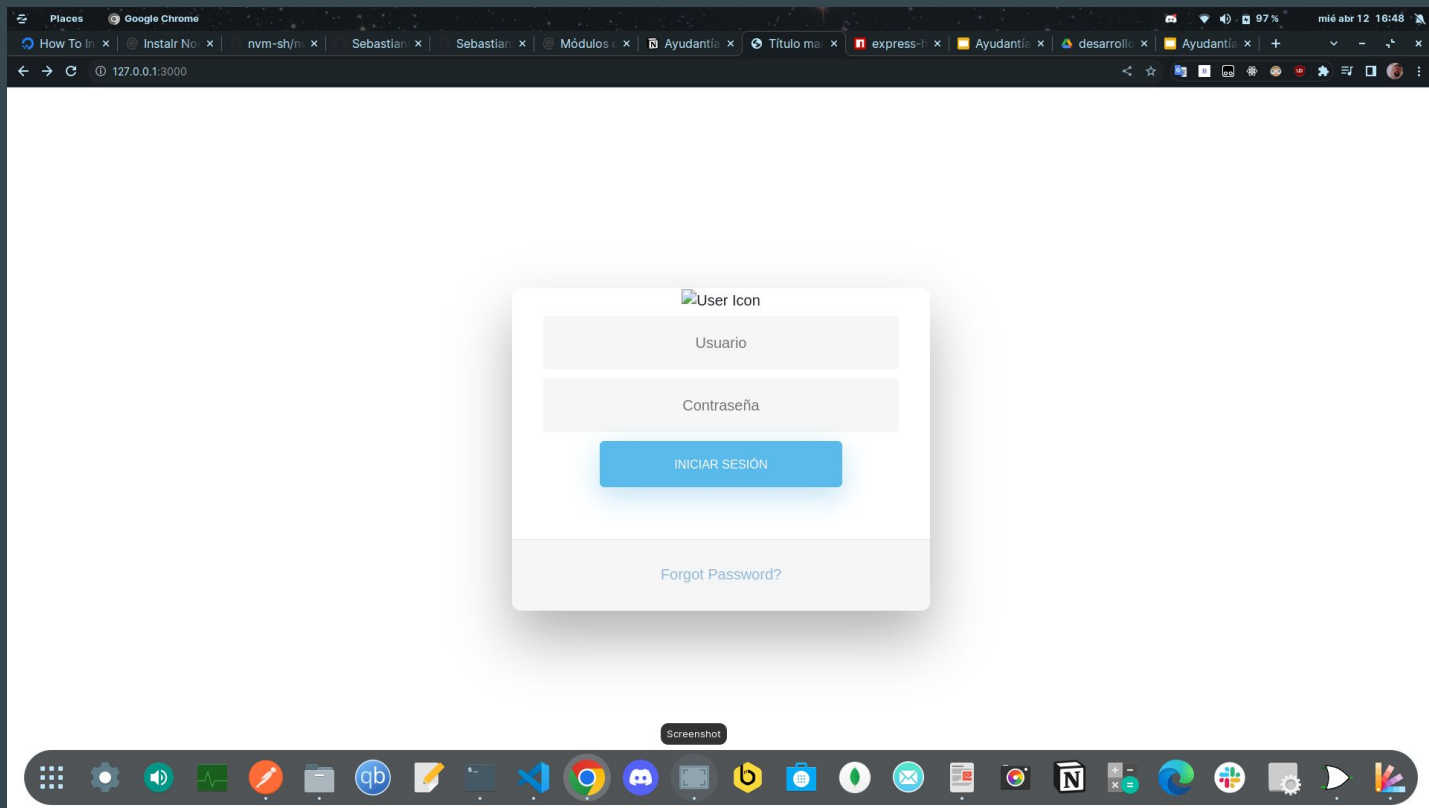
```
codigo_ayudantia3 > handlebars_y_login >  package.json >  author
1  {
2    "name": "express-handlebars-basic-example",
3    "version": "0.0.0",
4    "description": "",
5    "private": true,
6    "main": "server.js",
7    "type": "module",
8    > Debug
9    "scripts": {
10     "test": "echo \"Error: no test specified\" && exit 1",
11     "start": "node server.js"
12   },
13   "author": "",
14   "license": "BSD",
15   "dependencies": {
16     "express": "^4.17.1",
17     "express-handlebars": "^6.0.6"
18   }
19 }
```

Ejecutamos el proyecto para ver que todo esté bien



```
node server.js  
> ls  
'archivo configuracion Nginx'  package.json      server.js  
node_modules                  package-lock.json  views  
> node server.js  
Server express-handlebars corriendo en puerto: 3000
```

Visitamos IP:3000



server.js

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

```
main.handlebars U  package.json U  JS server.js U X
codigo_ayudantia3 > handlebars_y_login > JS server.js > ...
1  import express from "express";
2  import { engine } from 'express-handlebars'; // "express-handlebars"
3  //import res from "express/lib/response";
4
5  import * as path from "path";
6  import { fileURLToPath } from "url";
7  const __dirname = path.dirname(fileURLToPath(import.meta.url));
8
9  //console.log("__dirname: ", __dirname);
10
11  const app = express();
12
13  app.engine("handlebars", engine());
14
15  //Indico engine de mis views
16  app.set("view engine", "handlebars");
17
18  //Indico donde estan mis views
19  app.set("views", path.resolve(__dirname, "./views"));
20
21  // serve all other static files like CSS, images, etc
22  app.use(express.static(`${__dirname}/views`));
23
24
25  app.get("/", (req, res) => {
26    res.render("login");
27  });
28
```

Ejemplo de una view: login.handlebars

Este es nuestro login.handlebars:

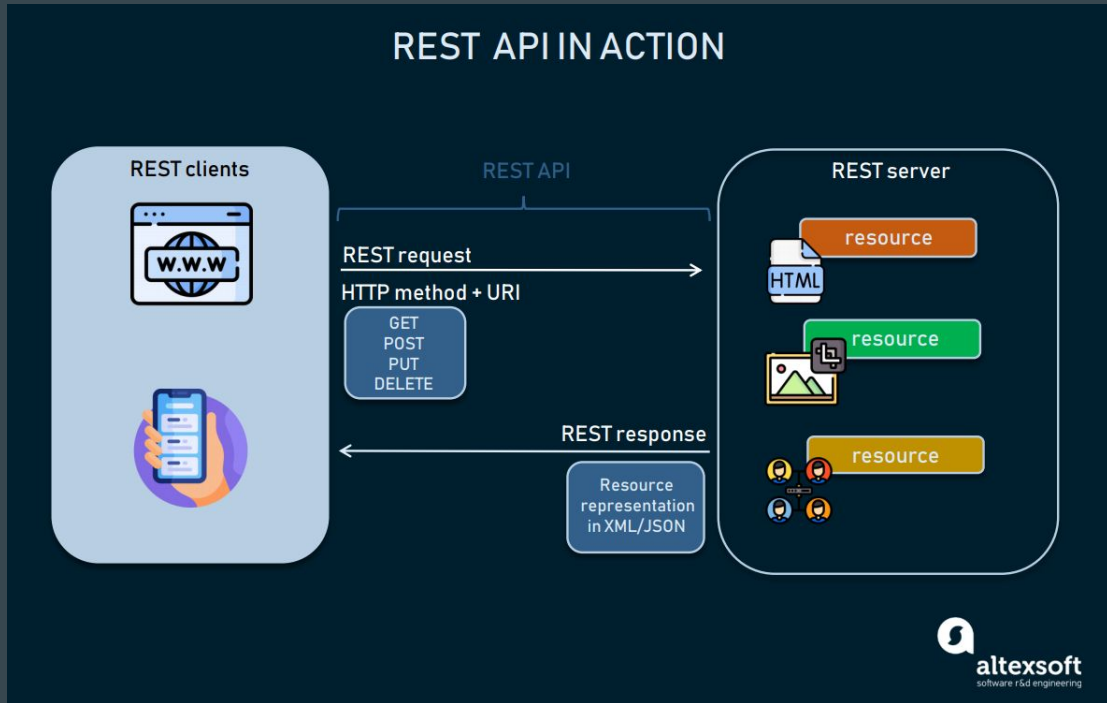
```
1 <div class="wrapper fadeInDown">
2   <div id="formContent">
3     <!-- Tabs Titles -->
4
5     <!-- Icon -->
6     <div class="fadeIn first">
7       
8     </div>
9
10    <!-- Login Form -->
11    <form action="/login" method="GET">
12      <input type="text" id="login" class="fadeIn second" name="user" placeholder="Usuario">
13      <input type="text" id="password" class="fadeIn third" name="pass" placeholder="Contraseña">
14      <input type="submit" class="fadeIn fourth" value="Iniciar sesión">
15      <p>{{fallido}}</p>
16    </form>
17
18    <!-- Remind Passowrd -->
19    <div id="formFooter">
20      <a class="underlineHover" href="#">Forgot Password?</a>
21    </div>
22
23    <style>...
287  </style>
288
289  </div>
290</div>
```

Y así cargamos el login.handlebars en nuestra app

```
25   app.get("/", (req, res) => {  
26     |     res.render("login");  
27   });  
28
```

¿Qué es un API?

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



Veamos cómo usarlo con nuestro proyecto

