

## Webgrafia


- ➔ Node: què és i per a què serveix > <https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/>
- ➔ Npm: què és i per a què serveix > <https://www.hostinger.es/tutoriales/que-es-npm>
- ➔ Express > <https://expressjs.com/es/>
- ➔ Node + Express > [https://developer.mozilla.org/es/docs/Learn/Server-side/Express\\_Nodejs/Introduction](https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction)
- ➔ Bones pràctiques en construir una API REST > <https://www.freecodecamp.org/news/rest-api-design-best-practices-build-a-rest-api/>
- ➔ Verbs HTTP > [https://es.wikipedia.org/wiki/Protocolo\\_de\\_transferencia\\_de\\_hipertexto](https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto)
- ➔ Connecting Express to MySQL > <https://codehandbook.org/connect-express-to-mysql/>
- ➔ Desenvolupament web amb NodeJS > <https://apuntes.de/nodejs-desarrollo-web/#gsc.tab=0>
- ➔ Com fer servir POSTMAN? > <https://desarrolloweb.com/articulos/como-usar-postman-probar-api>


## Com constuir una API REST amb NodeJS i Express?

### Pas 1: Descàrrega i instal·lació de NodeJS i de npm

- Anem a <https://nodejs.org>
- Descarreguem la versió de Llarg Suport (LTS) que trobem i que ens sortirà ja pensada per al sistema operatiu que estiguem fent servir. Això us ho aconsellem si esteu a Windows o a Mac OS.

Si treballem amb un Ubuntu el millor per instal·lar-se **NodeJS i npm**

 NO ÉS baixar-se directament de la web de node el programa ni tampoc instal·lar-lo del repositori d'Ubuntu mitjançant l'ordre apt install, sinó que la

 MILLOR MANERA és fer-ho mitjançant l'ordre curl sobre una web i, a continuació, fer la instal·lació mitjançant l'ordre apt install.

Aquí us deixem un [exemple](#) on s'hi explica com fer la instal·lació de diferents maneres però una d'elles és amb curl. Instal·larem **NodeJS** i de retruc també el gestor de paquets node **npm**.

- Instal·lem.

- Un cop hem instal·lat tot, comprovem que ho tenim tot correcte. Per fer-ho obrim la nostra terminal/console i escrivim

**node -v**  
(i també **npm -v**)

Ens han d'aparèixer les darreres versions de cadascun.

### Pas 2: Creació de l'entorn

- On vulguem, creem una carpeta de nom, per exemple, **api-rest**. Entrem a dins.
- Ara en la terminal (i dins de la nostra carpeta) escrivim

**npm init**

per crear el nostre primer arxiu **package.json**. Aquest serà com el nostre manifest de l'aplicació, i tot el que instal·lem estarà aquí definit.

### Pas 3: Instal·lació d'Express

Express és un framework de Nodejs que ens facilita la comunicació client – servidor amb http. És molt conegut i utilitzat.

- Ens trobem de nou dins de la nostra carpeta **api-rest**. Fem servir la línia de comandes per instal·lar aquesta paquet. Escrivim

**npm install express --save**

--save és per tenir escrit/guardat a l'arxiu package.json el que hem instal·lat

- Ara veiem que tenim una subcarpeta nova de nom **node\_modules** amb tot el que s'ha instal·lat.

## Pas 4: Ho!a món amb NodeJs i Express

- Creem ara dins de la nostra carpeta **api-rest** l'arxiu que en aquesta UF2 portarà tota la logística (farà de controlador i de model). Aquest arxiu es dirà **index.js**. L'obrim amb el nostre IDE i escrivim

```
'use strict'
const express = require('express')
const app=express()
app.listen(3000, ()=>{
  console.log('Aquesta és la nostra API-REST que corre en http://localhost:3000')
})
```

(**opcional**) Habitualment farem servir el port 3000. Si volem que això sigui el més genèric possible, un altra manera de fer ús del port seria

```
'use strict'
const express = require('express')
const app=express()
const port =process.env.PORT || 3000
app.listen(port, ()=>{
  console.log(`Aquesta és la nostra API-REST que corre en http://localhost:${port}`)
})
```

Un cop guardem les línies de codi, anem a provar el nostre backend. Obrim una terminal/consola dins de la carpeta **api-rest** i escrivim

**node index.js**

A la consola /terminal veurem com s'inicia el servidor pel port escollit (al nostre cas el 3000) i surt escrit la frase **Aquesta és la nostra API-REST que corre en http://localhost:3000**

## Pas 5: Configurar el nostre ApiRest

- Instal·lar el paquet **body-parser**. Aquest paquet ens serveix per a parsejar el cos de les peticions post. Per instal·lar-lo ens trobem dins de la nostra carpeta **api-rest**, i escrivim a la terminal/consola:

**npm i -S body-parser**

(això seria equivalent a **npm install body-parse --save**)

Un cop instal·lat, l'importem al nostre arxiu **index.js**

```
'use strict'

//importacions i creació de constants per a la seva utilització
const express = require('express')
const bodyParser=require('body-parser')
const app=express()

//configuració del bodyParser perquè admeti entrades json i
app.use(bodyParser.urlencoded({extended:false}))
app.use(bodyParser.json())
app.listen(3000, ()=>{
  console.log('Aquesta és la nostra API-REST que corre en http://localhost:3000')
})
```

- Instal·lem un paquet per no haver de parar i iniciar el nostre servidor per cada cosa que instal·lem i carreguem. Per fer-ho ens anem a la nostra carpeta **api-rest** i escrivim

**npm install --save-dev nodemon**

Si mirem el nostre arxiu **package.json** veurem les incusions fetes fins ara dels paquets instal·lats

Podríem haver fet aquesta darrera instal·lació en global (perquè serveixi per a totes les nostres aplicacions) però una manera més neta de fer-ho és un cop instal·lat **nodemon** fer al nostre **package.json** la següent inclusió en la categoria scripts:

```
"scripts": {
  "start": "nodemon index.js",
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

Ara per arrencar el nostre servidor només caldrà escriure a la terminal (i dins de la carpeta **api-rest**):

**npm start**

Per comprovar que ara tenim actualitzacions automàtiques, fent un canvi qualsevol a **index.js**, per exemple, escrivint el port 3001 en lloc de 3000, guardem i veiem com això s'hi aplica a l'instant.

## Pas 6: Fer servir els verbs

### 6.1 Get

```
'use strict'
const express = require('express')
const bodyParser=require('body-parser')
const app=express()
app.use(bodyParser.urlencoded({extended:false}))
app.use(bodyParser.json())

//només fem la petició get
app.get('/hola',(req,res)=>{
  res.send({message:'Hola món'})
})

//fem la petició get i recollim un paràmetre
app.get('/hola/:name', (req,res)=>{
  res.send({message: `Hola món ${req.params.name}!` })
})

app.listen(3000, ()=>{
  console.log('Aquesta és la nostra API-REST que corre en http://localhost:3000')
})
```

Per provar-ho, ens anem al nostre navegador i escrivim

[localhost:3000/hola](http://localhost:3000/hola)

i veurem que ens hi respon amb aquest json: `{message:'Hola món'}`

i ara escrivim

[localhost:3000/hola/Maria](http://localhost:3000/hola/Maria)

i veurem que ens hi respon amb aquest altre json: `{message: 'Hola món Maria!'}`

### 6.2 Post, Put, Delete

Seria bo d'instal·lar al nostre navegador una extensió per a visualitzar millor els json que farem servir. Per exemple a Chrome seria **Json Formatter**

Baixem o provem via web l'aplicació **Postman** per poder provar aquests verbs.

En aquest punt us recomano que llegiu a la Webgrafia un document que parla de com fer servir Postman per a provar APIs

## Pas 7: Ens connectem a una BBDD (MySQL)

Ara ens anem a connectar des de Node Express a una BBDD de MySQL.

Per poder fer-ho hem de tenir instal·lat com a mínim MySQL Server / Maria DB, o un paquet integrat de l'estil de LAMPP o XAMPP.

Un cop ho tenim tot instal·lat, hem de tenir una manera de comunicar-nos amb MySQL Server/ Maria DB. Això ho podem fer mitjançant PhpMyAdmin (que es troba als paquets integrats LAMPP o XAMPP) o [Mysql Workbench](#) (aplicació que podràs trobar i instal·lar des de la pàgina oficial de MySQL, i que ens deixa gestionar les BBDD de MySQL) o alguna eina universal per administrar una BBDD (per exemple, [DBeaver](#))

Un cop tenim tot el nostre entron de treball preparat, només ens queda crear una BBDD de MySQL de nom, per exemple, **testM06**, una taula de nom, per exemple, **users**, i dos camps: **username** (varchar 50 primary key) i **userpass** (varchar 150)

Tornem a l'arxiu **index.js** i fiquem el següent codi:

```

'use strict'

/////AIXÒ JA HO TENÍEM

const express = require('express')
const bodyParser=require('body-parser')
const app=express()
app.use(bodyParser.urlencoded({extended:false}))
app.use(bodyParser.json())

///// AIXÒ ÉS NOU I SERIA PER TREBALLAR AMB MYSQL
///// COMPTE: hem d'instal·lar mysql per a Node Express amb npm i -S mysql
///// importem mysql

const mysql = require('mysql');

///// declarem els paràmetres de connexió (millor si l'usuari de connexió no és root sinó un usuari específic per aquesta BBDD
///// i amb permissos restringits

var connection = mysql.createConnection({
    host: 'localhost',
    database: 'test',
    user: 'root',
    password: ''

});

///// fem servir la BBDD que tenim
app.post('/api/login', function (req, res) {
    console.log("estem a login");

    ///// provem de connectar-nos i capturar possibles errors

    connection.connect(function(err) {
        console.log(err);
        if (err) {
            console.error('Error connecting: ' + err.stack);
            return;
        }
        console.log('Connected as id ' + connection.threadId);
    });

    connection.query('SELECT * FROM users', function(error, results,field){
        if (error){
            res.status(400).send({resultats: null})
        }else{
            /////COMPROVACIÓ DE DADES PER CONSOLA DE NODE
            // console.log(results);
            // results.forEach(result => {
            // console.log(result.user);
            // })
            res.status(200).send({resultats: results})
        }
    });

})

app.listen(3000, ()=>{
    console.log('Aquesta és la nostra API-REST que corre en http://localhost:3000')
})

```