# 01 Primeros pasos

Vamos a comenzar con las instalaciones necesarias para configurar nuestro proyecto.

## Instalación Software

- https://nodejs.org/es/
- https://www.mongodb.com/download-center/community
- https://robomongo.org/
- https://code.visualstudio.com/
- https://expressjs.com/es/starter/installing.html

## Crear servidor

Comenzaremos con las configuraciones básicas de nuestro servidor.

1. Crear una carpeta en alguna parte de tu computador
2. Arrastrar dicha carpeta a Visual Studio Code
3. Ejecutar un nuevo proyecto de Node.js

```
npm init --yes
```

4. Instalar Express.js

```
npm install express --save
```

5. Crear archivo `app.js`

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

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

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

La aplicación inicia un servidor y escucha las conexiones en el puerto 3000. La aplicación responde con "Hello World!" para las solicitudes al URL raíz (/) o a la ruta raíz. Para cada vía de acceso diferente, responderá con un error 404 Not Found.

6. Ejecute la aplicación con el siguiente comando:

```
node app.js
```

7. A continuación, cargue http://localhost:3000/ en un navegador para ver la salida.
8. Para que detecte los cambios nuestro servidor podemos instalar https://www.npmjs.com/package/nodemon

```
npm install -g nodemon
```

9. Configurar nuevo script en package.json

```
"dev": "nodemon app.js"
```

Ahora ejecutar:

```
npm run dev
```

10. Asignar puerto automáticamente:

```
app.set('puerto', process.env.PORT || 3000);
app.listen(app.get('puerto'), function () {
  console.log('Example app listening on port'+ app.get('puerto'));
});
```

## Instalar Morgan (middleware)

Nos sirve para pintar las peticiones HTTP request que se solicitan a nuestro aplicación.

```
npm i morgan --save
```

```
const morgan = require('morgan');
app.use(morgan('tiny'));
```

## Instalar CORS (middleware)

Para realizar solicitudes de un servidor externo e impedir el bloqueo por CORS.

```
npm install cors --save
```

Enable All CORS Requests

```
const cors = require('cors');
app.use(cors());
```

# Instalar JSON y urlencoded (middleware)

Estos middlewares están disponibles en Express v4.16.0 en adelante.

```
app.use(express.json());

//application/x-www-form-urlencoded
app.use(express.urlencoded({ extended: true }))
```

# Static (middleware)

Incorporaremos un directorio público, por lo tanto puedes crear una carpeta `public` con un archivo `index.html` para mostrar un resultado, de igual forma configuraremos nuestro archivo `app.js`

```
// Para acceder al directorio actual
const path = require('path');

app.use(express.static(path.join(__dirname, 'public')));
```

# History Vue.js

Vue.js utiliza el modo History para simular las rutas de un sitio web, ya que al ser SPA es un simple HTML, esto nos puede traer problemas con Express por lo tanto agregaremos otro Middleware de configuración. https://router.vuejs.org/guide/essentials/history-mode.html#example-server-configurations

Este se llama connect-history-api-fallback

```
npm install --save connect-history-api-fallback
```

Muy importante copiar el siguiente código por abajo de la configuración de las rutas y dejar la configuración de rutas estáticas al final:

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

// Middleware para Vue.js router modo history
const history = require('connect-history-api-fallback');
app.use(history());
app.use(express.static(path.join(__dirname, 'public')));
```

## Resumen

En resumen nuestro archivo `app.js` debería ir quedando así:

```
const express = require('express');
const morgan = require('morgan');
const cors = require('cors');
const path = require('path');

const app = express();

// Middleware
app.use(morgan('tiny'));
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
//app.use(express.static(path.join(__dirname, 'public')));

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

// Middleware para Vue.js router modo history
const history = require('connect-history-api-fallback');
app.use(history());
app.use(express.static(path.join(__dirname, 'public')));

app.set('puerto', process.env.PORT || 3000);
app.listen(app.get('puerto'), function () {
```

```
  console.log('Example app listening on port'+ app.get('puerto'));
});
```

## Nodemon Babel

Instalar babel (pasar código de ES6 a ES5)

```
npm install -D @babel/core @babel/cli @babel/preset-env
@babel/node
```

Si te das cuenta, sigo cambiando de -Dy --save- estas banderas le dicen a npm si guardarlo como a devDependencyo como a dependency. Cuando finalice la instalación, ahora estamos listos para agregar nuestro script de desarrollo.

Crear archivo `.babelrc` en la raíz del sistema

```
{
  "presets": ["@babel/preset-env"]
}
```

Configurar Script en `package.json`

```
"scripts": {
  "dev": "nodemon app.js",
  "devbabel": "nodemon app.js --exec babel-node"
},
```

Ahora podemos pasar nuestro código a ES6 sin problemas:

```
import express from 'express';
import morgan from 'morgan';
import cors from 'cors';
import path from 'path';

const app = express();

// Middleware
app.use(morgan('tiny'));
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

```
// app.use(express.static(path.join(__dirname, 'public')));

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

// Middleware para Vue.js router modo history
const history = require('connect-history-api-fallback');
app.use(history());
app.use(express.static(path.join(__dirname, 'public')));

app.set('puerto', process.env.PORT || 3000);
app.listen(app.get('puerto'), () => {
  console.log('Example app listening on port'+ app.get('puerto'));
});
```

Prueba tu servidor ejecutando:

```
npm run devbabel
```