

## Índice.

1. Frameworks para node.
2. Electrón: pequeña introducción.
3. Creación de ejecutables con electrón.

En este tema veremos de una forma rápida lo que pretendemos crear durante este curso: aplicaciones de escritorio con interfaces gráficas de usuario usando tecnologías web, en concreto javascript, node y electrón. Crearemos un ejemplo pero no entramos en profundidad en muchos conceptos y quedarán muchas cosas en el aire. Las iremos viendo en otros temas con más detenimiento y quedarán despejadas las dudas.

### 1. Frameworks para javascript.

Un framework es un conjunto de software (librerías, funciones, módulos,...) que funcionan con un determinado lenguaje de programación. Existen frameworks para java, c, etc. Por supuesto también para node. Es conveniente aclarar que todo puede ser hecho sin frameworks y partir de cero, pero permiten que la programación sea más productiva.

Javascript tiene multitud de frameworks que facilitan la creación de aplicaciones de muchos tipos. Algunos muy usados:

- Express: permite crear servidores y servicios web.
- Ionic: para crear aplicaciones móviles híbridas.
- Electron: para desarrollar aplicaciones de escritorio que se puedan ejecutar en windows, OS o linux.
- Framework del lado cliente: para diseñar la parte cliente de una aplicación web, hay muchos: angular, vue, etc.

Por supuesto con node se pueden emplear todos estos frameworks. Muchos de estos frameworks pueden trabajar de manera conjunta, por ejemplo, hace un tiempo se popularizó el stack MEAN, acrónimo de mongoDb-express-angular-node. Con electron podemos usar también express y angular.

### 2. Electrón: pequeña introducción.

Electron lo que hace es envolver aplicaciones desarrolladas con tecnologías web para que se pueden ejecutar como app de escritorio multiplataforma. Es importante señalar **que casi**

**todo lo que haremos en este curso podría ejecutarse sin electron y serían aplicaciones web normales.**

Electrón se instala como un módulo mediante npm. Como todos los módulos de node, para trabajar con electron podemos usar dos vías: instalarlo globalmente en el ordenador o instalarlo localmente en la carpeta de un proyecto node (se instala en la carpeta node-modules. Vamos instalarlo de forma local, **dentro de nuestra carpeta de proyecto** se usa el comando:

```
npm install electron
```

Cuando usamos esta manera de instalarlo electron no se instala en el ordenador globalmente, sino que se instala dentro de la carpeta de nuestro proyecto, concretamente en el subdirectorio node-modules. Además el comando anterior crea una entrada de dependencia en el package.json del proyecto:

```
“dependencies” : “electron x.x.x”
```

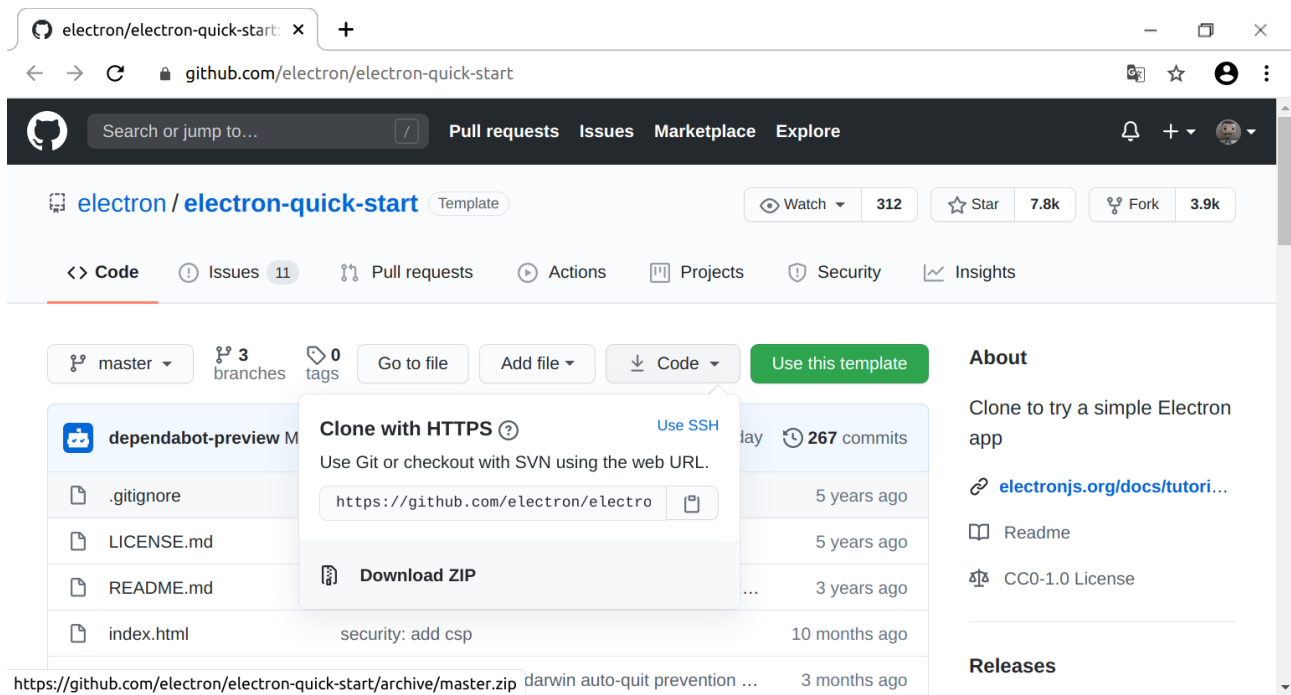
Trabajar de forma local tiene múltiples ventajas. Una de ellas es que cada uno de nuestros proyectos podrá estar hecho en una versión distinta de electrón. Además, si queremos llevarnos de un PC a otro nuestro proyecto node, lo mejor es copiar todo el proyecto menos la carpeta node-modules (que tenderá a ser grande). Para reconstruir la carpeta node-modules usaremos, desde la carpeta del proyecto, el comando: npm install

### **3. Creación de aplicaciones de ejecutables con electrón.**

Vamos a crear un ejecutable con node y electrón, un “hola mundo”. Lo primero que haremos será bajarnos desde github un proyecto mínimo con electron:

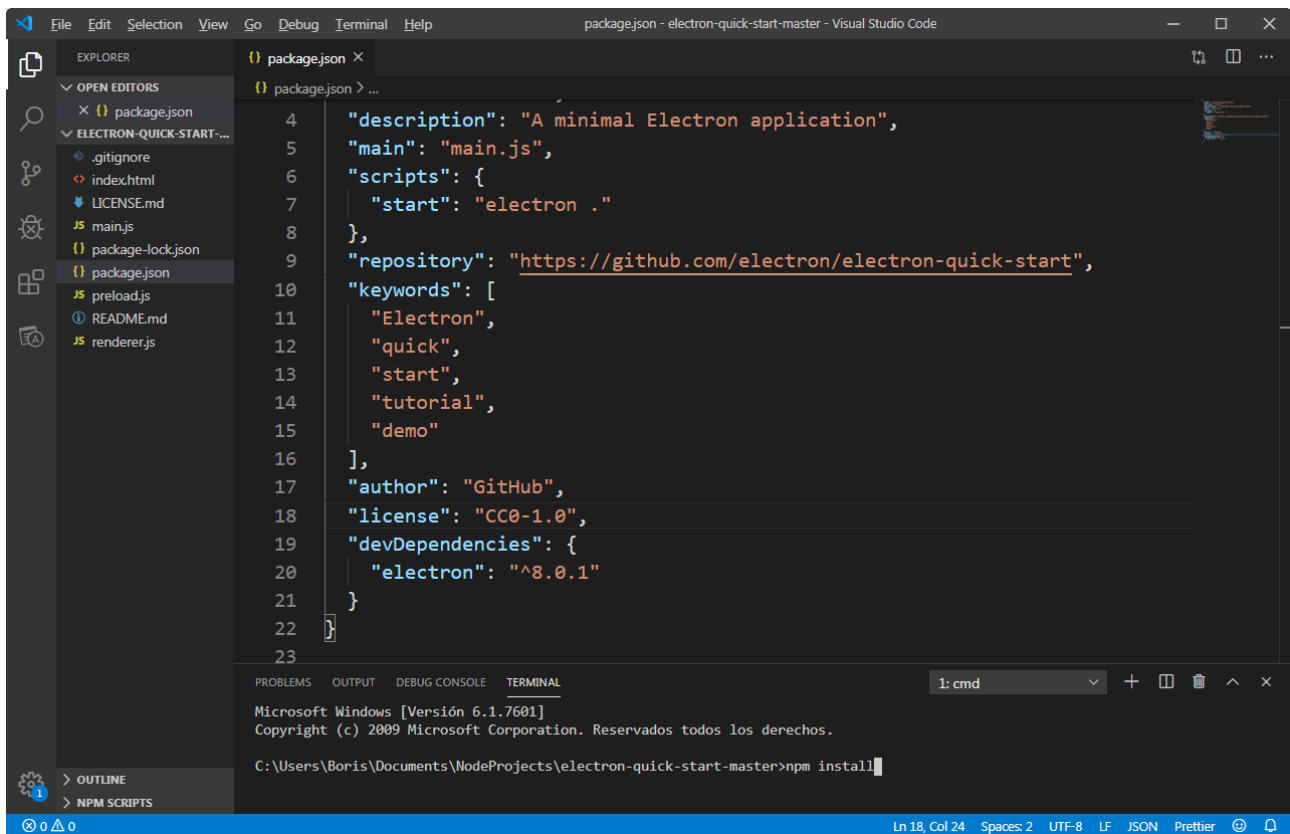
<https://github.com/electron/electron-quick-start>

Lo bajamos en zip:



Haciendo clic en Code y Download ZIP.

Lo descomprimos y abrimos la carpeta con code.

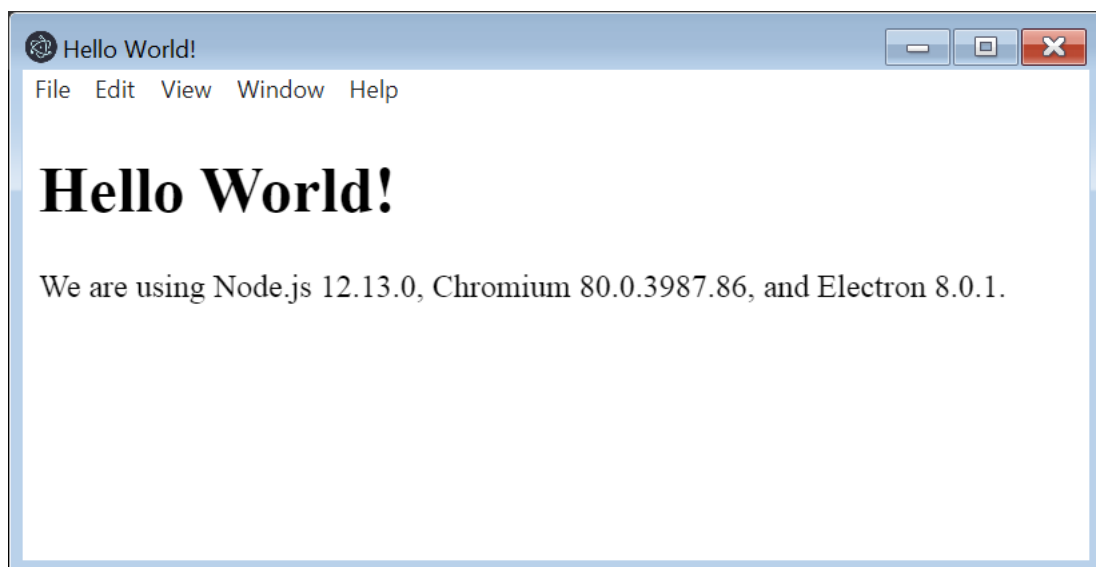


The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: package.json, .gitignore, index.html, LICENSE.md, main.js, package-lock.json, package.json (selected), preload.js, README.md, and renderer.js. The main editor displays the content of package.json:

```
4  "description": "A minimal Electron application",
5  "main": "main.js",
6  "scripts": {
7    "start": "electron ."
8  },
9  "repository": "https://github.com/electron/electron-quick-start",
10 "keywords": [
11   "Electron",
12   "quick",
13   "start",
14   "tutorial",
15   "demo"
16 ],
17 "author": "GitHub",
18 "license": "CC0-1.0",
19 "devDependencies": {
20   "electron": "^8.0.1"
21 }
22
23
```

At the bottom, the Terminal panel shows the command prompt with the command `npm install` entered. The status bar at the bottom indicates the current position is Line 18, Column 24, with 2 spaces, UTF-8 encoding, LF line endings, and JSON syntax highlighting.

Crea varios archivos. Por ahora no vamos a entrar en cada uno de ellos, los veremos detenidamente en temas posteriores. Si nos vamos a fijar en package.json. Como se puede ver está creada una entrada para electron como dependencia. Si ejecutamos el comando `npm install`, lo instalará en node-modules. También tiene una entrada `start` para ejecutar el proyecto, lo hacemos con `npm start`:



Para generar un ejecutable usaremos electron-packager, instalándolo (siempre dentro de nuestra carpeta de proyecto):

```
npm install electron-packager
```

Además de instalar el módulo crea una entrada en package.json:

```
"electron-packager": "^14.2.1"
```

Ahora incluiremos una entrada en para ejecutar este paquete desde npm:

```
"scripts": {  
  "start": "electron .",  
  "build": "electron-packager ."  
},
```

(Para poder ejecutar electron-packager en windows es necesario tener instalado .net framework 4.5 o superior y powershell 3 o superior, esto es porque vamos a generar un ejecutable o archivo .exe y lo que hará será compilar el proyecto para c++)

Lo ejecutamos con:

```
npm run build
```

Esto creará un directorio dependiendo de la arquitectura y sistema operativo, en mi caso:

```
electron-quick-start-win32-x64
```

Dentro podemos encontrar el ejecutable:

```
electron-quick-start.exe
```