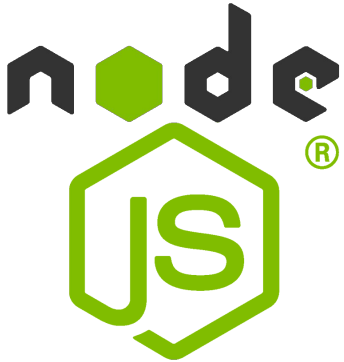
Introducción

¿Qué herramientas vamos a utilizar hoy?

- NodeJS
- NPM
- Git
- Electron-forge



NodeJS

NodeJS es un entorno de ejecución de javascript que nos permite crear aplicaciones de red escalables (como servidores), pero nosotros lo vamos a utilizar para crear una aplicación de escritorio.

NodeJS

Aunque se pueden utilizar IDEs para trabajar en nodeJS, nosotros vamos a utilizar un editor de texto (vscode) y un terminal para hacerlo más universal.

Electron

Desarrollado por GitHub. Electron nos permite crear aplicaciones de escritorio utilizando tecnologías ya existentes para el desarrollo web. Además, las aplicaciones desarrolladas por electron son multiplataforma. Para ejecutar la aplicación en cualquier entorno, electron se aprovecha del motor de renderizado de chromium.

Electron vs Java

Ventajas de Electron:

- Fácil de aprender.
- Aplicaciones bonitas fácilmente.
- librerías de Javascript.

Ventajas de Java:

- Mayor rendimiento.
- Aplicaciones que se sienten nativas.
- Librerías de Java.
- Amplia documentación.

Electron

Aplicaciones populares creadas con Electron:

Atom, Visual Studio Code, Slack, Microsoft Teams, Github Desktop, Whatsapp Desktop, Figma Desktop...

Empecemos

Descargar NodeJS: usuarios de Windows

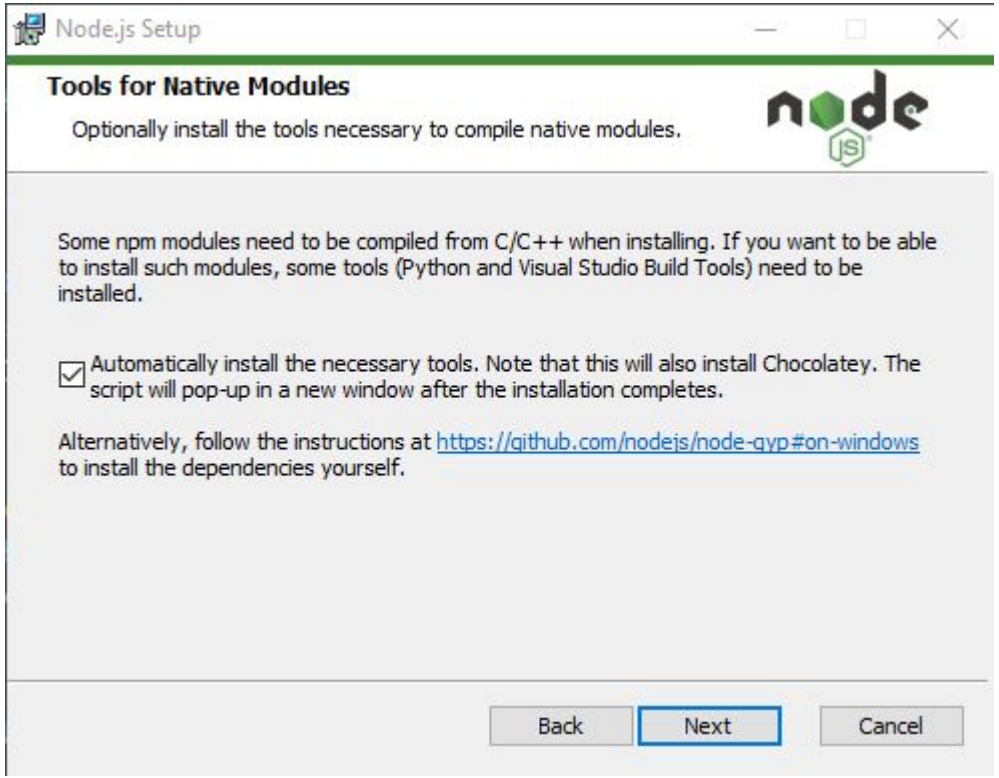
Podeis descargar el instalador desde su página web:

https://nodejs.org/es/download/

Descargar NodeJS: usuarios de Windows

Es importante que al instalar nodeJS dejemos marcada la opción *“install all the necessary tools.”*

Más tarde es posible que usar chocolatey para instalar git si no lo teneis instalado (Git Bash no cuenta).



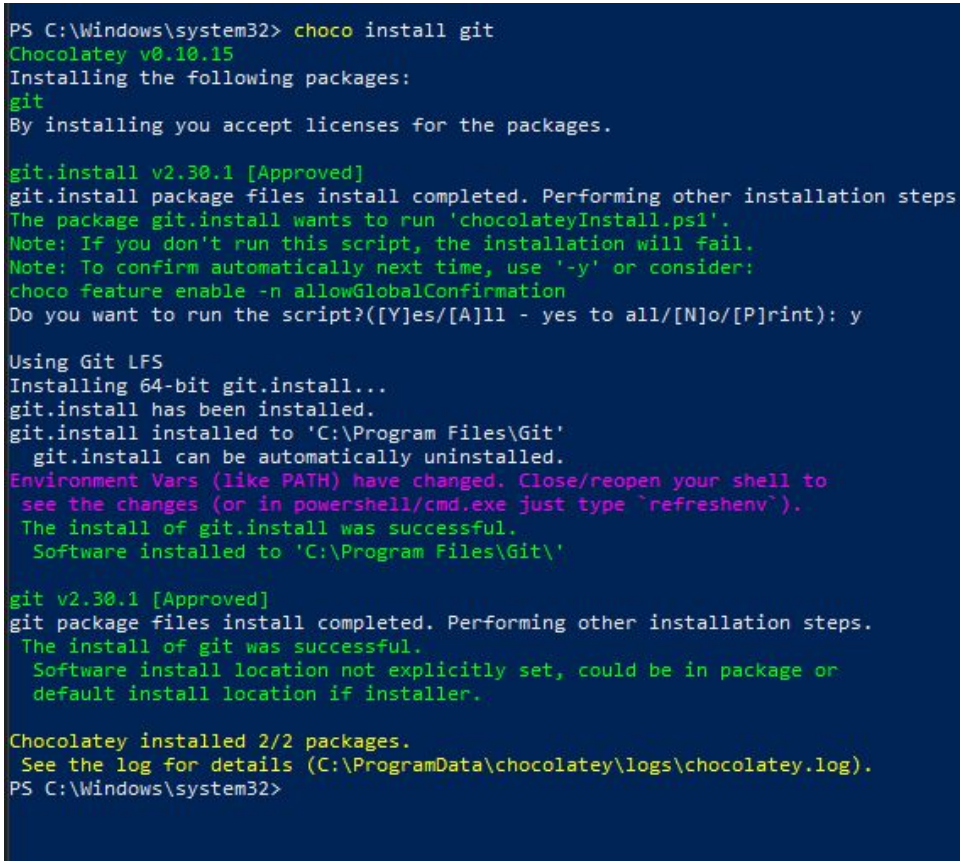
Descargar NodeJS: usuarios de Windows

Para instalar git en nuestro ordenador podemos abrir un terminal como administrador (WIN+X y seleccionarlo) y escribir el comando

choco install git

Seguramente descargar git de aquí también funcione, pero no he probado:

https://git-scm.com/download/win



Descargar NodeJS: usuarios de Linux

- Debian/Ubuntu/Derivados: sudo install nodejs npm
- Fedora: sudo dnf install nodejs npm
- Arch: sudo pacman -S nodejs npm

Descargar NodeJS: usuarios de MAC

Link: https://nodejs.org/es/download/

A partir de ahí seguro que puedes tú solo. Ánimo, yo confío en tí ;)

Creando un proyecto de Electron

En un terminal vamos a posicionarnos en la carpeta en la que queremos crear el proyecto y ejecutamos el siguiente comando:

npx create-electron-app my-new-app

Los que prefiráis yarn en lugar de NPM y sepáis lo que estáis haciendo podeis usar:

yarn create electron-app my-new-app

Error típico: Git

La plantilla creará un repositorio de Git. Podríamos copiar la plantilla a mano pero cualquier proyecto que se precie va a utilizar Git en cualquier caso. Si no tenéis instalado Git os encontrareis un error.

```
PS C:\Users\Knigh\Documents\electron> npx create-electron-app my-app
√ Initializing Project Directory
× Initializing Git Repository

An unhandled error has occurred inside Forge:
Command failed: git init
programa o archivo por lotes ejecutable.

Error: Command failed: git init
"git" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

    at ChildProcess.exithandler (child_process.js:308:12)
    at ChildProcess.emit (events.js:315:20)
    at maybeClose (internal/child_process.js:1048:16)
    at Process.ChildProcess._handle.onexit (internal/child_process.js:288:5)
PS C:\Users\Knigh\Documents\electron> choco install git
```

package.json

El archivo **package.json** contiene información relativa a nuestro proyecto como el nombre, la descripción, las instrucciones de compilación y las dependencias del proyecto.

index.js

No lo vamos a tocar mucho. Su método principal es *createWindow*, en el cual añadiremos ciertos parámetros como su tamaño por defecto o la presencia de la barra de menús y la consola.

```
const createWindow = () => {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 500,
    height: 600,
  });

  mainWindow.setMenuBarVisibility(false)

  // and load the index.html of the app.
  mainWindow.loadFile(path.join(__dirname, 'index.html'));

  // Open the DevTools.
  mainWindow.webContents.openDevTools();
};
```

Creando una app

Creamos un nuevo archivo app.js y añadimos el siguiente código:

```
let isShowingImage = false;

async function downloadImage() {
  const ID = 'image'
  if (isShowingImage) {
    document.body.removeChild(document.getElementById(ID));
  }
  const URL = 'https://dog.ceo/api/breeds/image/random';
  const response = await fetch(URL);
  const data = await response.json();
  const image = document.createElement('img');
  image.setAttribute('id', ID);
  image.src = data.message;
  document.body.appendChild(image);
  isShowingImage = true;
}

document.getElementById('button').addEventListener('click', downloadImage);
```

Creando una app

Añadimos el script a index.html y creamos un botón que llame al script:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>GENERADOR DE PERROS</title>
    <link rel="stylesheet" href="index.css">
    <script src='prueba.js'></script>
  </head>
  <body>
    <h1>GENERADOR DE PERRETES</h1>
    <button id="startBtn" class="comenzar" onClick="downloadImage()">DAME UN PERRO QUIERO UN PERRO</button>
    <br>
  </body>
</html>

```

Testeando nuestra app

Para ejecutar nuestra app y comprobar su funcionamiento sólo tenemos que posicionarnos en la carpeta correcta en el terminal y ejecutar el comando:

npm start

Compilando nuestra app.

Por defecto, Node incluye muchas advertencias. Estas advertencias son muy útiles en desarrollo. Sin embargo, estas hacen nuestra app más pesada y lenta, así que debes asegurarte de usar la versión de producción cuando despliegues la aplicación.

Compilando nuestra app.

Para compilar nuestra app utilizamos el comando:

```
npm run make
```

Este comando se asegurará de realizar el compilado y el empaquetado de la aplicación correctamente. Por defecto, sólo se realizará el empaquetado para nuestro sistema operativo.

Compilando nuestra app.

Para compilar nuestra app en plataformas que no sean la nuestra vamos a editar nuestro archivo package.json:

```
"scripts": {
  "start": "electron-forge start",
  "package": "electron-forge package",
  "make-win": "electron-forge make --platform win32",
  "make-mac": "electron-forge make --platform darwin",
  "make-linux": "electron-forge make --platform linux",
  "publish": "electron-forge publish",
  "lint": "echo \"No linting configured\""
},
```

Tras esta modificación utilizaremos el comando npm run make-{win,mac,linux} para generar los paquetes correspondientes.

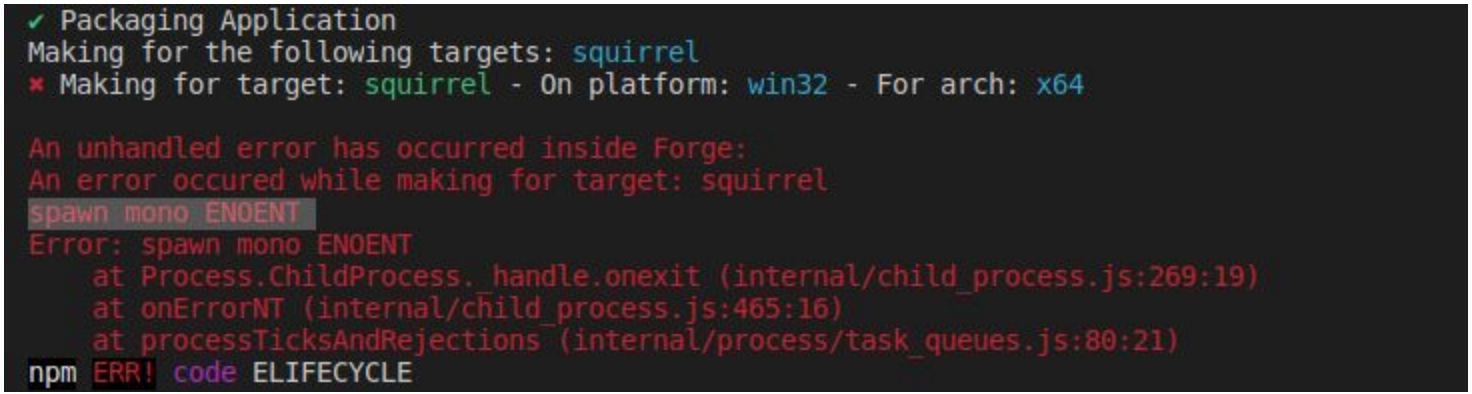
Error típico: Linux

Estando en un entorno linux, necesitas tener instalado tanto dpkg como rpm para poder compilar en ambos formatos. En caso contrario encontrarás un error que te indicará lo que falta.

```
> electron-forge make

✔ Checking your system
✔ Resolving Forge Config

An unhandled rejection has occurred inside Forge:
Error: Cannot make for deb, the following external binaries need to be installed: dpkg, fakeroot
    at MakerDeb.ensureExternalBinariesExist (/home/rafa/Documents/electron/GDGApp2/node_modules/@electron-forge/maker-base/src/
Maker.ts:147:13)
    at _default (/home/rafa/Documents/electron/GDGApp2/node_modules/@electron-forge/core/src/api/make.ts:148:11)
    at /home/rafa/Documents/electron/GDGApp2/node_modules/@electron-forge/cli/src/electron-forge-make.ts:44:5

Electron Forge was terminated. Location:
{}
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! gdgapp2@1.0.0 make: `electron-forge make`
```

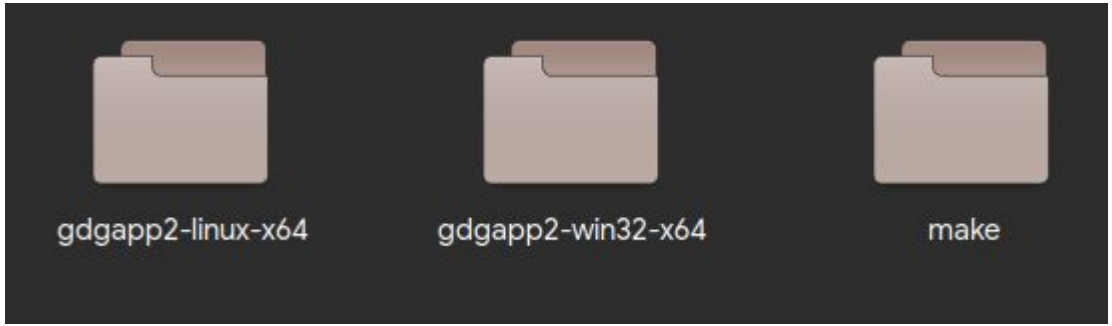
Error típico: Linux

Estando en un entorno linux, necesitas tener instalados los paquetes **wine** y **mono** para generar los archivos de windows.



Compilando nuestra app.

Una vez realizadas las compilaciones, se habrá creado una carpeta **out** en nuestro proyecto donde están nuestros archivos.



Dentro de la carpeta make se encontrarán los instaladores de las apps que hemos compilado.

¿PREGUNTAS?