

NOMBRE DEL PROYECTO
DETALLES
DETALLES

Instalación y ejecución de Electron

Orientador
DR.Ramón Antonio Alvarez López

Autor
Andrés David Dávila Arroyo

2021

Índice

1. FRAMEWORK	3
1.1. USO DE LOS FRAMEWORK EN INTERNET	3
2. EL FRAMEWORK ELECTRON	3
2.1. ¿QUÉ ES CHROMIUM?	4
2.2. ¿QUÉ ES NODE.JS?	4
2.2.1. ¿CÓMO FUNCIONA NODE.JS?	4
2.2.2. ¿PARA QUÉ SIRVE NODE.JS?	4
2.2.3. ¿QUÉ ES Y PARA QUÉ SIRVE NPM?	5
3. MANEJO DE VERSIONES	5
3.1. ¿QUE ES GIT Y GITHUB?	6
3.1.1. GIT	6
3.1.2. FLOJO DE TRABAJO DE GIT	7
3.1.3. GITHUB	7
4. INSTALACIÓN DE VISUAL ESTUDIO CODE	8
5. INSTALACIÓN DE NODE.JS	12
6. CREACIÓN DE UN PROGRAMA SENCILLO UTILIZANDO NODE.JS	17
6.1. CONCEJOS A SEGUIR	17
6.1.1. VIDEO EDUCATIVO	17
7. INSTALACIÓN DE ELECTRON Y EJECUCIÓN DE UN PROGRAMA DE ESCRITORIO SENCILLO, USANDO VISUAL STUDIO CODE y NODE.JS	19
7.1. CONCEJOS A SEGUIR	19
7.1.1. CURSO EDUCATIVO	19
7.2. INICIO DE LA ACTIVIDAD PRINCIPAL	20
Referencias	30

Índice de figuras

1.	Flojo de trabajo GIT	7
2.	DESCARGA DE VISUAL CODE	8
3.	TÉRMINOS DE LICENCIA	9
4.	UBICACIÓN INSTALACIÓN	9
6.	EXTENSIONES	10
7.	PÁGINA OFICIAL	12
8.	EJEMPLO NODE.JS	17
9.	CURSO ELECTRON.JS	19
10.	CARPETA CREADA	20
11.	ABRIENDO CARPETA DESDE VISUAL CODE	20
12.	ARCHIVO PACKAGE.JSON	23
13.	CLIC EN CREAR CARPETA Y DESPUÉS ASIGNAMOS NOMBRE	24
14.	CARPETAS CREADAS	24
15.	INSTALACIÓN DE ELECTRON	25
16.	CONFIRMACIÓN INSTALACIÓN	25
17.	CARPETA NODE_MODULE	26
18.	PAQUETE DE ELECTRON	26

CAPITULO I

Durante este capítulo estaré hablando acerca del **Framework Electron** y su principal uso en la actualidad. Identificaremos cuáles son las herramientas necesarias para instalarlo y ejecutarlo en nuestra computadora.

1. FRAMEWORK

El Framework es una especie de plantilla, un esquema conceptual, que simplifica la elaboración de una tarea, ya que solo es necesario complementarlo de acuerdo a lo que se quiere realizar. (Gabriela, 2021)


1.1. USO DE LOS FRAMEWORK EN INTERNET

Para cualquier proyecto en Internet se requiere un desarrollador web que produzca el software o la aplicación que necesitamos. Dependiendo del tipo de proyecto, esta tarea puede durar mucho tiempo si se crea de la nada. Es necesario elaborar parte por parte, haciendo pruebas y aciertos hasta conseguir el objetivo. Todo esto puede requerir uno o más programadores, además del tiempo suficiente para realizar las pruebas necesarias hasta que el software esté funcionando perfectamente. Sin embargo, los Framework permiten entregar un proyecto en menos tiempo y con un código más limpio, cuya eficacia ya ha sido comprobada. A partir del Framework los programadores pueden complementar y/o modificar la estructura base para entregar el software o la aplicación que cumpla los objetivos requeridos. (Gabriela, 2021)

2. EL FRAMEWORK ELECTRON

Electron es un framework para crear aplicaciones de escritorio usando **JavaScript**, **HTML** y **CSS**. Incrustando **Chromium** y **Node.js** dentro del mismo, Electron le permite mantener una base de código JavaScript y crear aplicaciones multiplataforma que funcionan en **Windows**, **macOS** y **Linux**, no requiere experiencia en desarrollo nativo. Miles de organizaciones que abarcan todas las industrias utilizan Electron para construir Software multiplataforma. (Electron, s.f.)

Alguna de las aplicaciones más famosa creadas por electron son:

★ Visual Studio Code 

★ Facebook Messenger 

★ Twitch 

2.1. ¿QUÉ ES CHROMIUM?

Es una base de código abierto para desarrollar un navegador web, mantenida por diversas compañías que posteriormente usan el código fuente para crear su propia versión de navegador con características adicionales. Además, es multiplataforma, ya que funciona con Windows, Linux, Mac y Android. Asimismo, cuenta con soporte para las extensiones de Chrome Store. ([Chromium](#), 2021)

2.2. ¿QUÉ ES NODE.JS?

Node.js, es un entorno en tiempo de ejecución multiplataforma para la capa del servidor (en el lado del servidor) basado en JavaScript. Este entorno es controlado por eventos diseñados para crear aplicaciones escalables, permitiéndote establecer y gestionar múltiples conexiones al mismo tiempo. Gracias a esta característica, no tienes que preocuparte con el bloqueo de procesos, pues no hay bloqueos. ([Simões](#), 2021)

2.2.1. ¿CÓMO FUNCIONA NODE.JS?

El diseño de Node.js está inspirado en sistemas como el Event Machine de Ruby o el Twisted de Python. Sin embargo, Node.js presenta un bucle de eventos como una construcción en tiempo de ejecución en lugar de una biblioteca. Este bucle de eventos es invisible para el usuario.

Otra característica especial de Node.js es que está diseñado para simplificar la comunicación. No tiene subprocesos, pero te permite aprovechar múltiples núcleos en su entorno y compartir sockets entre procesos. ([Simões](#), 2021)

2.2.2. ¿PARA QUÉ SIRVE NODE.JS?

Puedes utilizar Node.js para diferentes tipos de aplicaciones. Los siguientes son algunos de los ejemplos:

- ★ Aplicaciones de transmisión de datos (streaming)
- ★ Aplicaciones intensivas de datos en tiempo real
- ★ Aplicaciones vinculadas a E/S
- ★ Aplicaciones basadas en JSON:API
- ★ Aplicaciones de página única ([Simões](#), 2021)

Node.js también se integra dentro de la gestión de paquetes utilizando la herramienta NPM (Node Package Manager; esta se instala por defecto con Node.js), la cual nos permite instalar numerosos componentes a través de un repositorio en línea. Ejemplo del uso de Node.js [Clic aquí para ver ↵](#)

2.2.3. ¿QUÉ ES Y PARA QUÉ SIRVE NPM?



De sus siglas **NPM (Node Package Manager)** es un gestor de paquetes desarrollado en su totalidad bajo el lenguaje JavaScript por Isaac Schlueter, a través del cual podemos obtener cualquier librería con tan solo una sencilla línea de código, lo cual nos permitirá agregar dependencias de forma simple, distribuir paquetes y administrar eficazmente tanto los módulos como el proyecto a desarrollar en general.(Muradas, 2021)

3. MANEJO DE VERSIONES

Antes de hablar de **Gil** y **GitHub**. Primero tenemos que comprender el concepto de control de versiones y porque puede ser útil para tu investigación. En términos generales, un control de versiones consiste en tomar instantáneas de tus archivos a lo largo del proceso de creación. La mayoría de personas, de hecho, trabajan con algún sistema de control de versiones para gestionar sus archivos. A menudo, el control tiene lugar guardando distintas versiones de un mismo archivo. Por ejemplo, no es raro encontrarnos ante un directorio que contiene los siguientes archivos:(Strien, 2017)

midocumento.txt

midocumentoversion2.txt

midocumentoconrevisiones.txt

midocumentofinal.txt

Esta forma de nombrar los archivos puede ser más o menos sistemática. Si añadimos fechas, puede ser un poco más fácil seguir los cambios:

midocumento2016-01-06.txt

midocumento2016-01-08.txt

Aunque este método sea un poco más claro, sigue habiendo problemas. En primer lugar, este método no registra o describe qué cambios se han producido entre uno y otro archivo guardado. Pueden ser pequeñas correcciones de erratas, o bien tratarse de la reescritura de pasajes enteros o incluso de una modificación mayor, por ejemplo, de la estructura del documento. Además si quieres revertir alguno de estos cambios, tendrás que averiguar cuándo se hizo el cambio y deshacerlo.(Strien, 2017)

Con un control de versiones se persigue solucionar este tipo de problemas mediante la puesta en marcha de un registro sistemático de cambios en los archivos. A grandes rasgos, puede afirmarse que el control de versiones realiza instantáneas de los archivos a lo largo del tiempo. Estas instantáneas documentan el momento en que fueron tomadas, pero también qué cambios tuvieron lugar entre cada una de ellas, lo cual permite recuperar una versión más antigua de tu archivo. A partir de aquí se abre un sinfín de posibilidades gracias al control de versiones.(Strien, 2017)

En resumen con un controlador de versiones podemos hacer lo siguiente:

- ★ Rastrear el desarrollo y los cambios de tus documentos
- ★ Registrar los cambios que has hecho de una manera que puedas entender posteriormente
- ★ Experimentar con versiones distintas de un documento al mismo tiempo que conservas la más antigua
- ★ Fusionar dos versiones de un documento y administrar los conflictos existentes entre distintas versiones
- ★ Revertir cambios y volver atrás gracias al historial de versiones anteriores de tu documento

En concreto, el control de versiones es útil para facilitar la colaboración. De hecho, una de las razones que explican el origen del control de cambios es que permitiera a varias personas trabajar al mismo tiempo en un proyecto de considerables dimensiones. Para poder hacer esto trabajaremos con GIT y GITHUB(Strien, 2017)

3.1. ¿QUE ES GIT Y GITHUB?

3.1.1. GIT

GIT es un software de control de versiones, su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos (También puedes trabajar solo no hay problema). (*Introducción a GIT | Tutorial GIT / GITHUB*, s.f.)

3.1.2. FLOJO DE TRABAJO DE GIT

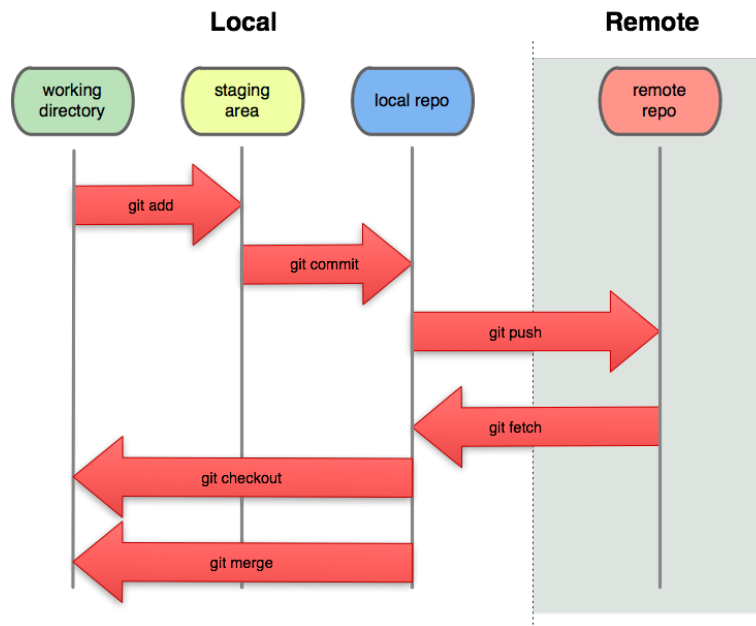




Figura 1: Flojo de trabajo GIT

Tratando de explicar la imagen: Tenemos nuestro directorio local (una carpeta en nuestro pc) con muchos archivos, Git nos irá registrando los cambios de archivos o códigos cuando nosotros le indiquemos, así podremos viajar en el tiempo retrocediendo cambios o restaurando versiones de código, ya sea en Local o de forma Remota (servidor externo). En la práctica quedará más claro. (*Introducción a GIT | Tutorial GIT / GITHUB, s.f.*)


Página oficial para que descargues este Software:  **Clíc aquí** ↗URL8↖


Videtutorial de intalacion:  **Ver video** ↗URL9↖

Tutorial web de intalacion:  **Clíc aquí** ↗URL10↖

3.1.3. GITHUB

Es una plataforma de desarrollo colaborativo para alojar proyectos (en la nube) utilizando el sistema de control de versiones Git, Además cuenta con una herramienta muy útil que es GitHub Pages donde podemos publicar nuestros proyectos estáticos (HTML, CSS y JS) gratis. (*Introducción a GIT | Tutorial GIT / GITHUB, s.f.*)

Página oficial para que entres y te registres:  **Clíc aquí** ↗URL11↖

Tutorial web de registro:  **Clíc aquí** ↗URL12↖

DESARROLLO




INICIACIÓN DEL PROCESO

Los Software, que estaremos utilizando son multiplataforma y se pueden instalar en Mac, Windows y Linux.

Las actividades que estaremos realizando durante en este capítulo son:

1. Instalar el Software Visual Studio Code
2. Instalar el Software Node.js
3. Realizar un ejemplo básico con Node.js
4. Instalar el Framework Electron y ejecutar un programa sencillo desde Visual Studio Code
5. Empaquetar nuestra aplicación y convertirla en un ejecutable .exe

4. INSTALACIÓN DE VISUAL ESTUDIO CODE

1. Para instalar **Visual Studio code**, nos dirigiremos a la página oficial  **Clic aquí**  **URL** . Descargaremos la versión correspondiente a nuestro sistema operativo, simplemente damos clic y se descargará de inmediato.

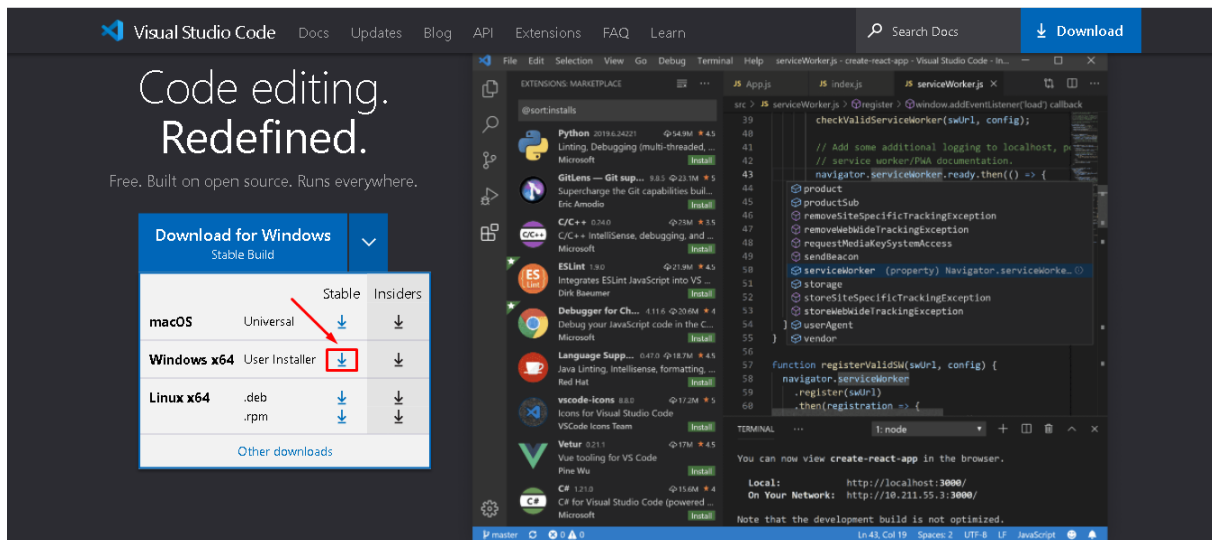


Figura 2: DESCARGA DE VISUAL CODE

2. Abrimos el ejecutable y aceptamos **los términos de licencia** y pulsamos **siguiente**.

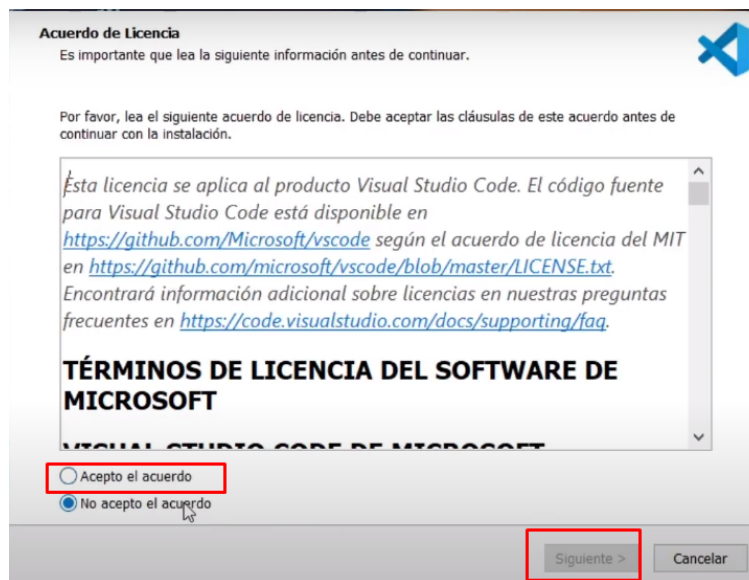


Figura 3: TÉRMINOS DE LICENCIA

3. Elegimos el destino de instalación y pulsamos **siguiente**.

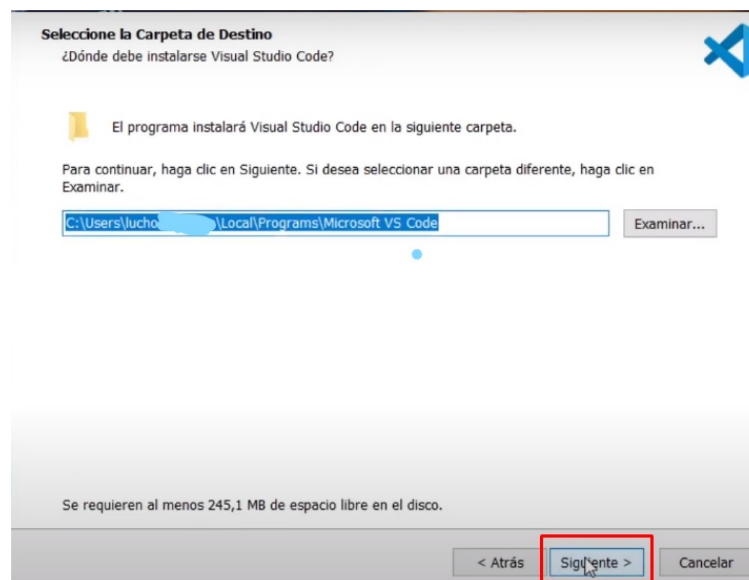


Figura 4: UBICACIÓN INSTALACIÓN

4. Pulsamos **siguiente** a todo lo que venga y por último esperemos que termine de instalar



5. Una vez instalado **Visual Studio**, lo abrimos e instalaremos una extensión que nos servirá para colocarlo en español. Damos clic en **extensiones**.

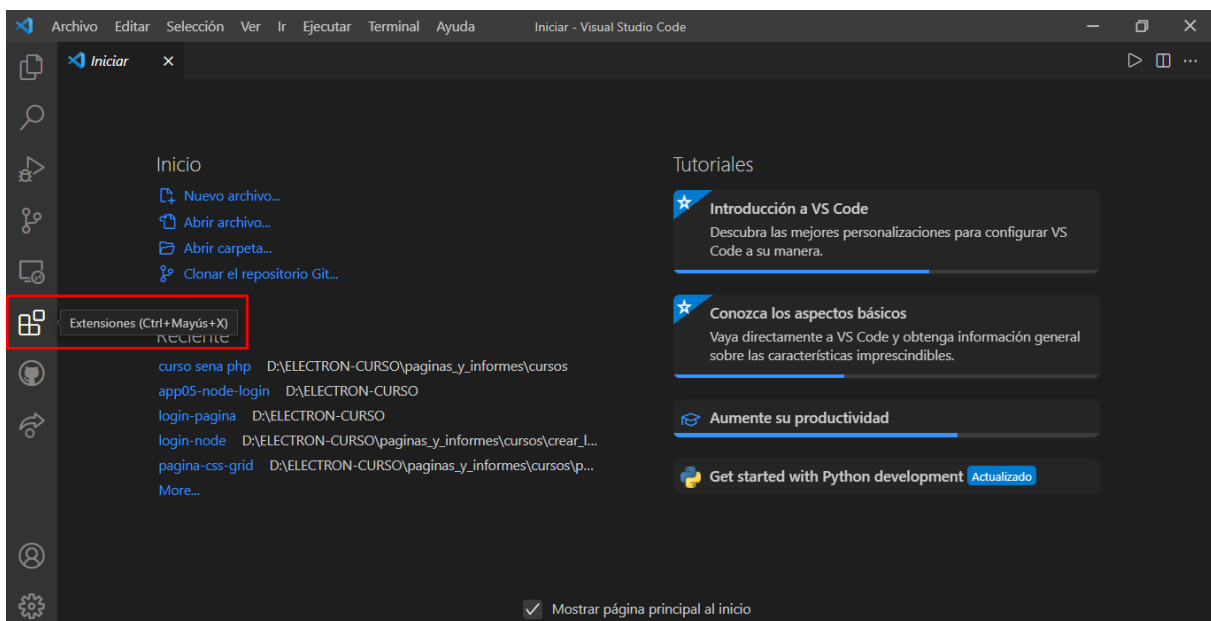
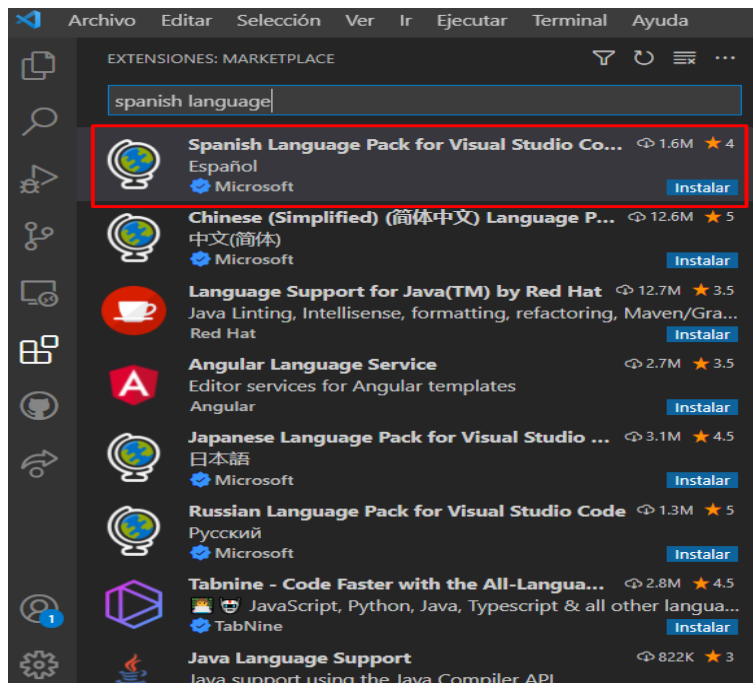


Figura 6: EXTENSIONES

6. En la barra de búsqueda escribiremos **Spanish Language** damos clic en la primera opción y después **pulsamos instalar** y con esto ya tendremos instalado Visual Studio Code y configurado en Español.



Algunas de las extensiones más importantes para **Visual Studio Code**, tenemos a:

- **vscode-icons**: nos ayudará a distinguir carpetas y tipos de archivos
- **live server**: esta extensión nos permitirá lanzar un servidor local de desarrollo
- **html snippets**: nos sirve para autocompletar código HTML
- **prettier - Code formatter**: es un formateador de código obstinado. Aplica un estilo coherente al analizar su código y volver a imprimirlo con sus propias reglas que tienen en cuenta la longitud máxima de línea, ajustando el código cuando es necesario.

5. INSTALACIÓN DE NODE.JS

1. Para instalar **Node.js** nos vamos primero a la página oficial de Node.js, [Clic aquí](#) **↗URL5↖**. Estando dentro de la página descargaremos la versión correspondiente a nuestro sistema operativo. Hasta la fecha: **16/11/2021**, la última versión es: **17.1.0**, pero es recomendable descargar la versión: **16.13.0**. Por lo tanto, esta será versión que descargaremos.

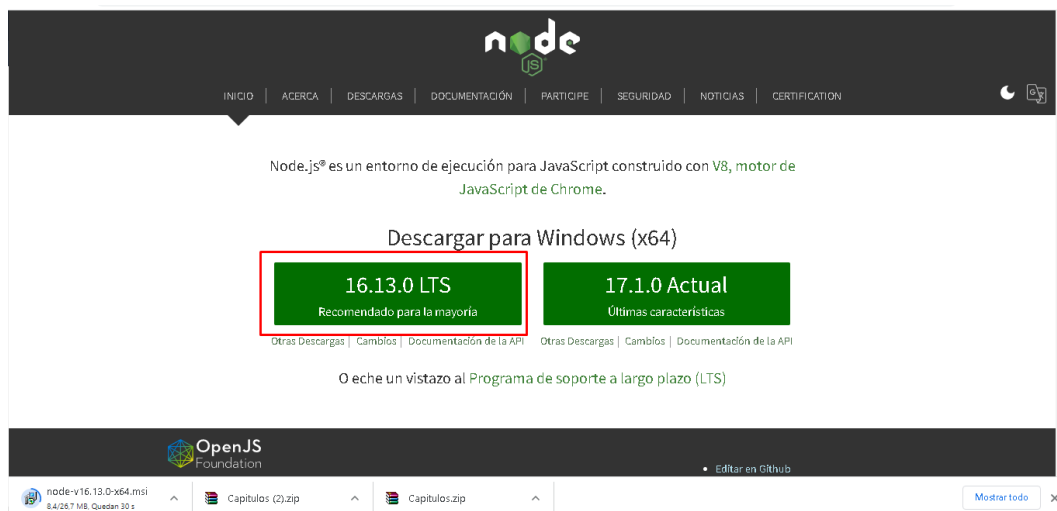
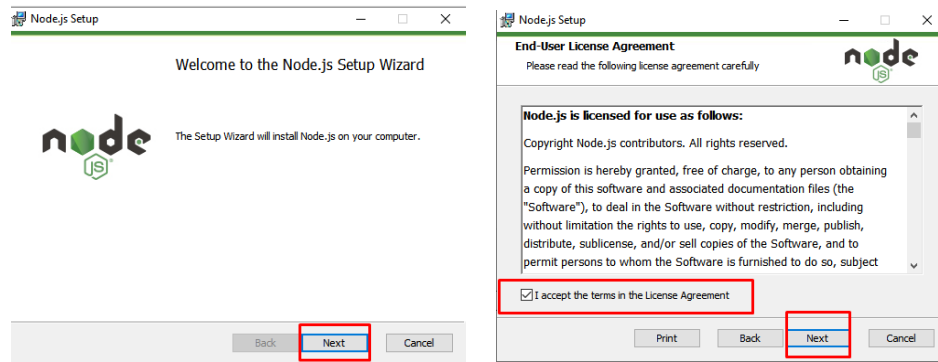
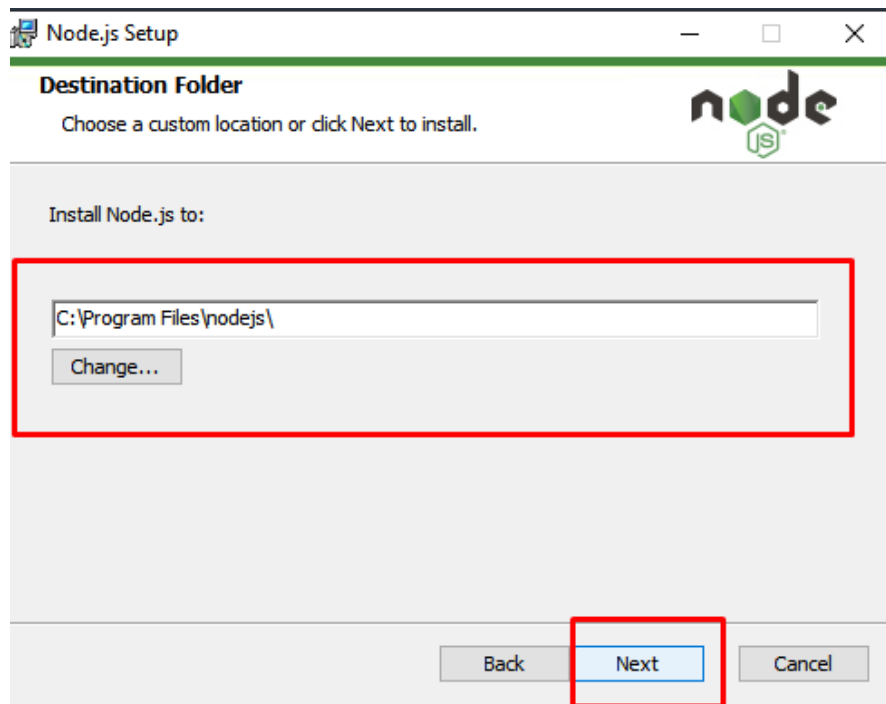


Figura 7: PÁGINA OFICIAL

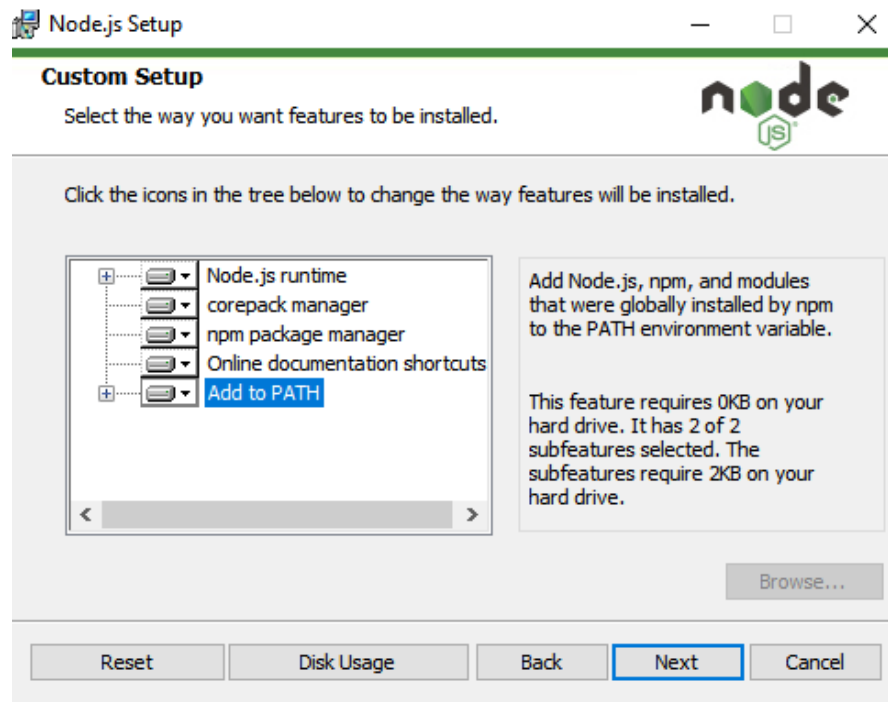
2. Abrimos el ejecutable, damos clic en **Next**, aceptamos los **términos de licencia** y pulsamos de nuevo **Next**



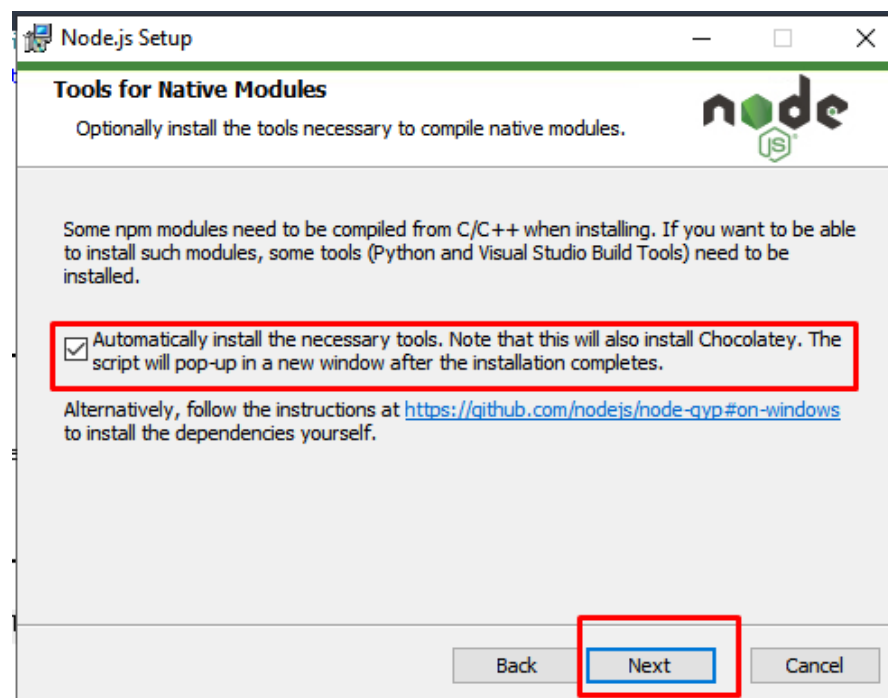
3. Ahora elegimos la ruta de instalación y pulsamos next



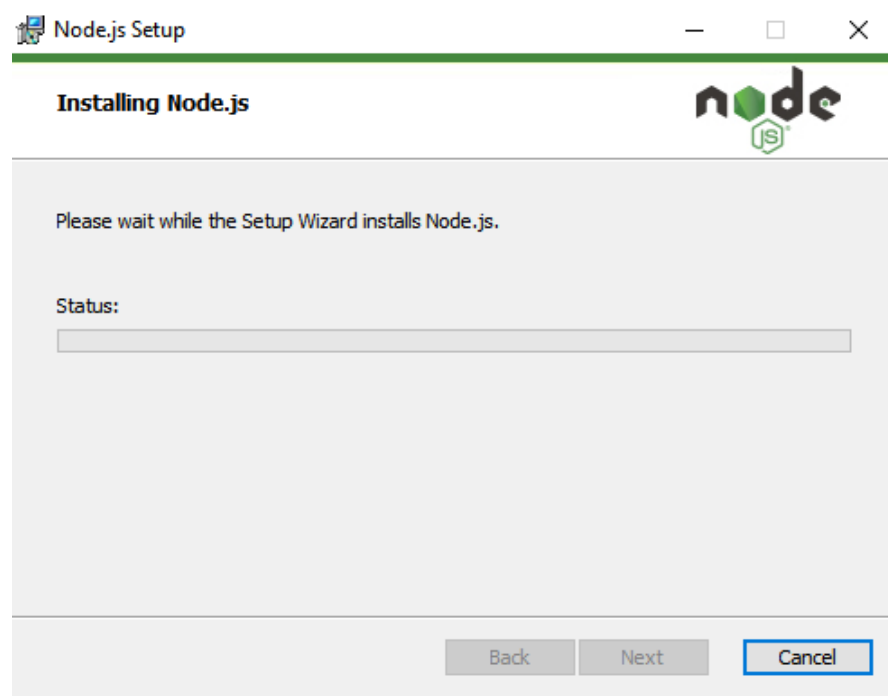
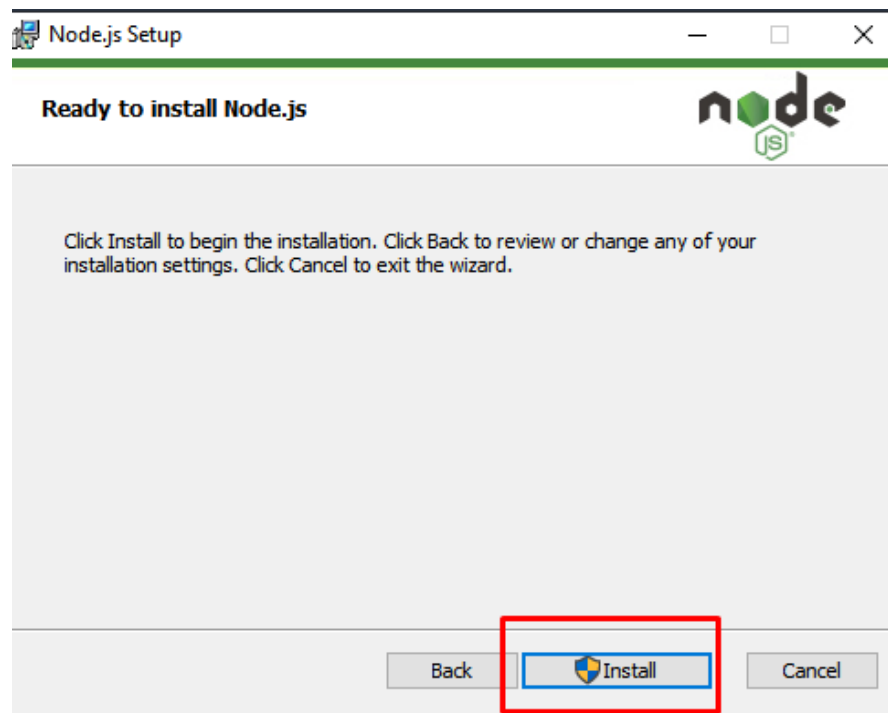
4. Elegimos los paquetes a instalar en este caso yo lo dejaré por defecto



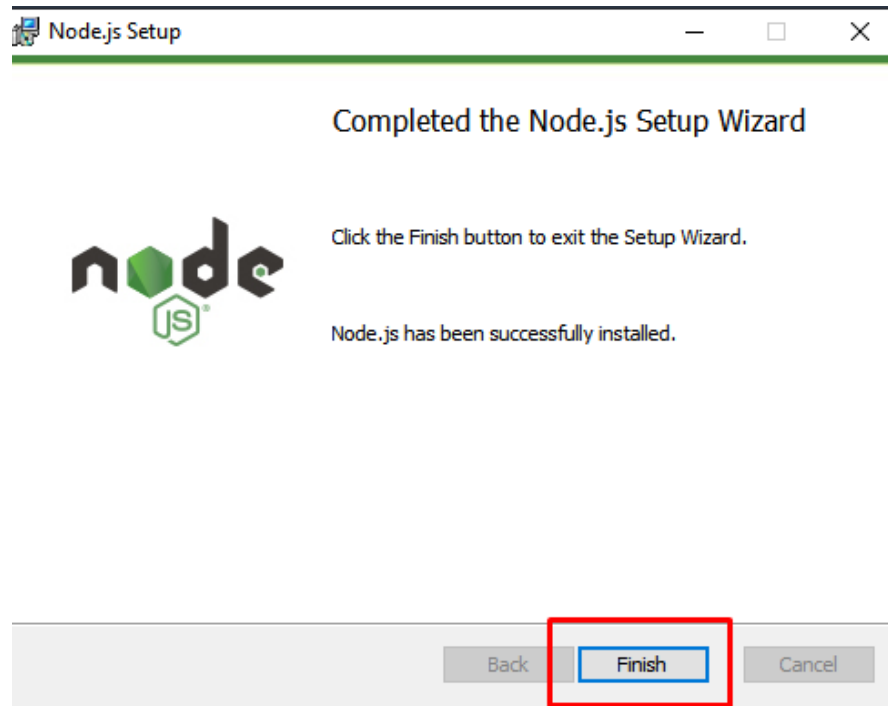
5. Nos ofrecerá instalar herramientas para compilar módulos nativos, por lo tanto, lo chuleamos y pulsamos **next**



6. Por último le daremos **Install** y esperemos que instale



7. Después que termine de instalar le damos **Finish**



8. Para comprobar la instalación nos vamos a nuestra consola y escribiremos el siguiente comando: **node -v** .Nos debería aparecer la versión que tenemos instalada de Node.js (**en mi caso la versión 16.13.0**). Para comprobar que se nos ha instalado también los paquetes NPM, escribiremos **npm -v** y pulsaremos de **nuevo Enter**. Con esto ya hemos terminado la instalación de Node.js

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.1237]
(c) Microsoft Corporation. Todos los derechos reservados.
C:\Users\USUARIO>node -v
v16.13.0
C:\Users\USUARIO>npm -v
8.1.0
C:\Users\USUARIO>
```

6. CREACIÓN DE UN PROGRAMA SENCILLO UTILIZANDO NODE.JS

Para poder realizar este ejercicio necesitarás primero tener instalado los Software **Node.js** [ver instalación](#) y el editor de código **Visual Studio Code** [ver instalación](#)

6.1. CONCEJOS A SEGUIR

Si quieres entender un poco más acerca de lo que haremos en esta actividad, te recomiendo que mires el video que se adjuntará a continuación.

6.1.1. VIDEO EDUCATIVO

Se adjunta código QR y link de acceso al video

Para escanear la imagen, te recomiendo la siguiente app: Lector de código QR.

Se adjunta link:  [↗URL1↖](#)



Figura 8: EJEMPLO NODE.JS

Link de acceso al video

A. Node.js  [Ver video](#) [↗URL3↖](#)

aquí va la explicación

7. INSTALACIÓN DE ELECTRON Y EJECUCIÓN DE UN PROGRAMA DE ESCRITORIO SENCILLO, USANDO VISUAL STUDIO CODE y NODE.JS

Para poder **Instalar Electron** y comenzar a desarrollar aplicaciones de escritorio utilizando este **Framework**, necesitarás tener instalado los Software **Node.js** y el editor de código **Visual Studio Code**.

7.1. CONCEJOS A SEGUIR

Si quieres entender un poco más acerca de lo que haremos en esta actividad y de paso aprender más sobre Electron, te recomiendo que el siguiente curso que se adjuntará a continuación.

7.1.1. CURSO EDUCATIVO

Se adjunta código QR y link de acceso al curso de electron

Para escanear la imagen, te recomiendo la siguiente app: Lector de código QR.




Se adjunta link:  [↗URL1↖](#)






Figura 9: CURSO ELECTRON.JS

Link de acceso al curso de electron

A. Electron.js  **Ver video** [↗URL2↖](#)

Si deseas saber más sobre el **Framework Electron**, te dejaré el link de la página oficial,  **Clic aquí**  **URL6** . Aquí encontrarás toda la información y documentación necesaria para trabajar con este Framework

7.2. INICIO DE LA ACTIVIDAD PRINCIPAL

Para realizar esta actividad nos guiaremos de un tutorial que nos brinda la propia página oficial de Electron:  **Clic aquí**  **URL7** .

1. Crearemos una carpeta en nuestro escritorio, se le puede colocar el nombre que queramos. Ahora nos dirigimos a nuestro editor de código Visual Studio Code y abriremos esa carpeta

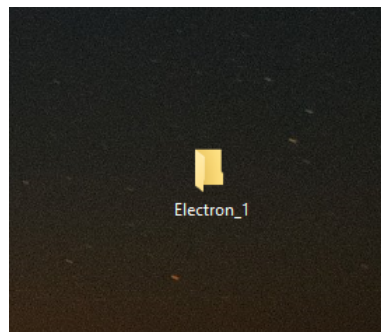


Figura 10: CARPETA CREADA

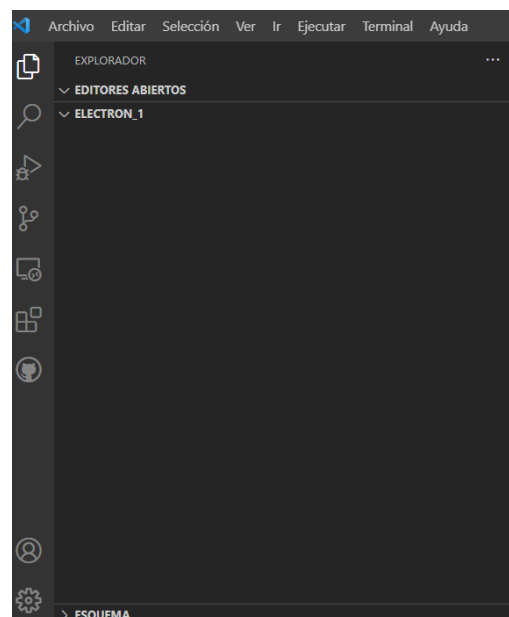


Figura 11: ABRIENDO CARPETA DESDE VISUAL CODE

ESTRUCTURA BASE

2. Dentro de esta carpeta vamos a estar creando toda la estructura base que se encargara de la ejecución de nuestra aplicación electron.

Dentro de nuestra carpeta estaremos creando los siguientes archivos:

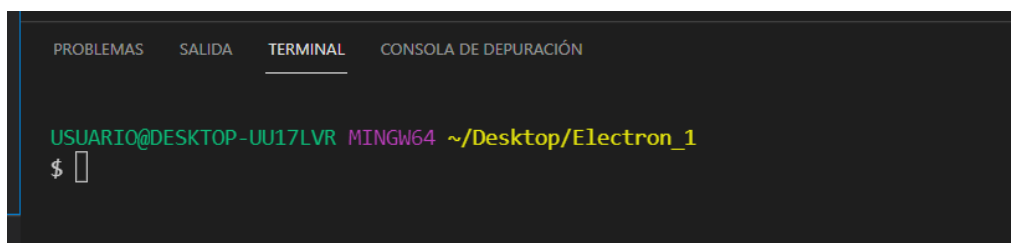
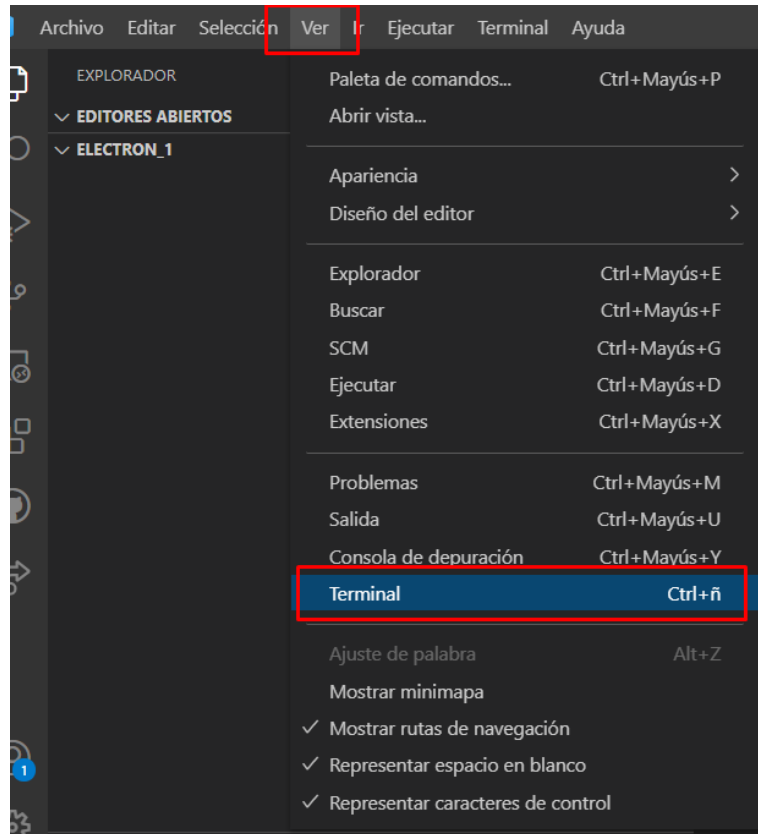
- a) **package.json**: este es generado automáticamente cuando se instalan paquetes o dependencias **npm** en el proyecto. Su finalidad principal es mantener todo el historial de los paquetes instalados y optimizar la forma en que se generan las dependencias del proyecto.
- b) **main.js**: este archivo será el punto de entrada del proyecto el que se encargara de arrancar el proceso principal de electron.
- c) **index.html**: nos mostrará la vista de inicio de la aplicación. Podemos también crear archivos adicionales como **CSS**, **JavaScript**. permitiéndonos darle estructura, estilos e interactividad a nuestra principal.

Al final nuestro directorio lucirá de la siguiente manera:

```
your-app/  
├package.json  
├main.js  
└index.html
```

CREACIÓN DE LOS ARCHIVOS

3. Para crear el archivo **package.json** primero nos dirigimos a la opción **ver**, luego a **terminal**, esto nos integrará una terminal en nuestro proyecto, ubicándose en la ruta actual.



4. Dentro de esta terminal escribiremos el siguiente comando: **npm init --yes** y pulsamos **Enter**. Esto nos creará el archivo **package.json**

```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN

USUARIO@DESKTOP-UU17LVR MINGW64 ~/Desktop/Electron_1
$ npm init --yes
```

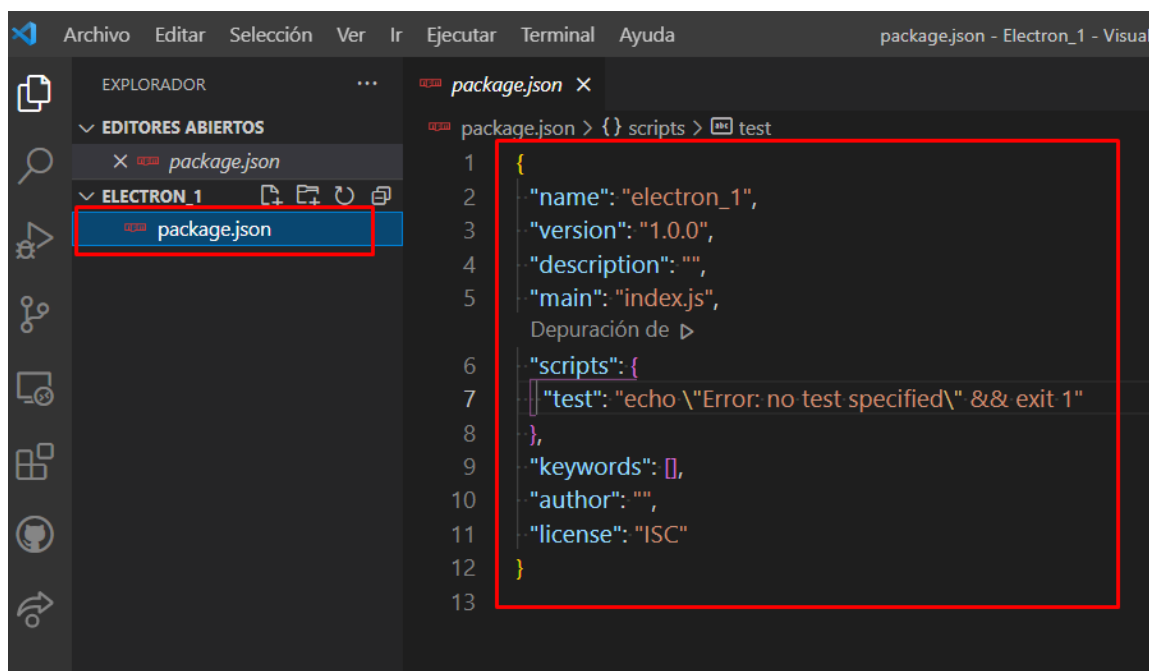


Figura 12: ARCHIVO PACKAGE.JSON

Campos más habituales de package.json:

- ★ **name:** nombre del proyecto, librería o paquete.
- ★ **version:** versión del paquete.
- ★ **description:** descripción breve del paquete o proyecto.
- ★ **main:** punto de entrada del proyecto. Suele ser index.js (node) o index.html (browser).
- ★ **scripts:** colección de scripts del proyecto. Aquí colocaremos los comandos que nos ayudaran a correr nuestro programa durante la fase de desarrollo.
- ★ **keywords:** palabras clave relacionadas con el proyecto.

- ★ **author:** nombre del autor del paquete.
 - ★ **license:** tipo de licencia del paquete o proyecto. las veremos más adelante cuando se comiencen a instalar **paquetes npm**
 - ★ **dependencies:** colección de paquetes para producción y la versión instalada. Estas son obligatorias en nuestro proyecto ya que son parte de la lógica de tu aplicación y son necesarias para que el proyecto funcione correctamente en producción
 - ★ **devDependencies:** colección de paquetes para desarrollo y la versión instalada. No son necesarias en producción y tu aplicación puede funcionar si ellas.
5. Ahora para tener más organizado el proyecto crearemos un directorio de nombre **src**, dentro de este se encontrará toda la estructura de proyecto, por lo tanto, en **src** crearemos una carpeta de nombre **views** aquí colocaremos todos los archivos **JS**, **HTML**, **CSS**, **IMÁGENES** necesarios para darle estructura, estilos e interactividad a la aplicación de escritorio.

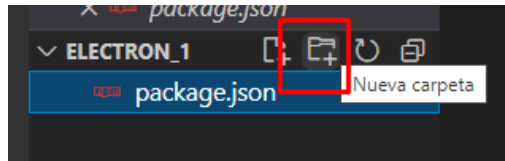


Figura 13: CLIC EN CREAR CARPETA Y DESPUÉS ASIGNAMOS NOMBRE

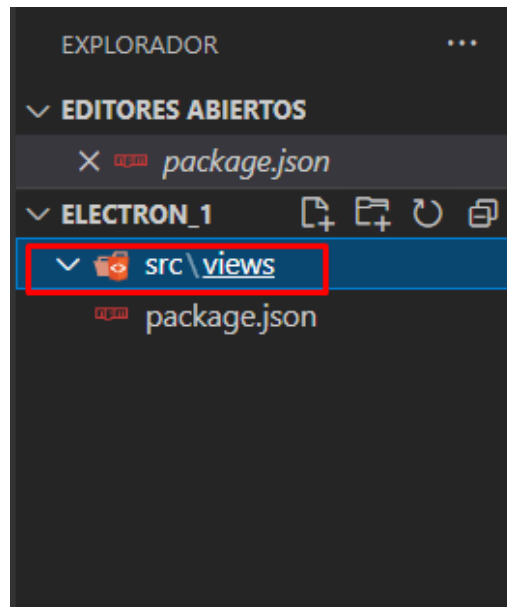
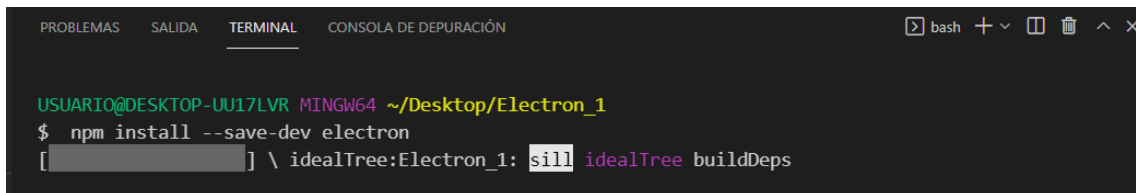


Figura 14: CARPETAS CREADAS

6. Ahora lo que haremos será instalar **Electron** como **devDependencies**. Para eso primero nos dirigimos a la terminal de comando que tenemos integrada en nuestro proyecto y digitamos lo siguiente: **npm install --save-dev electron** y pulsamos **Enter** y esperamos que instale Electron.



```
PROBLEMAS  SALIDA  TERMINAL  CONSOLA DE DEPURACIÓN
USUARIO@DESKTOP-UU17LVR MINGW64 ~/Desktop/Electron_1
$ npm install --save-dev electron
[ ] \ idealTree:Electron_1: sill idealTree buildDeps
```

Figura 15: INSTALACIÓN DE ELECTRON

7. Si la instalación se realizó correctamente en nuestro archivo **package.json** nos aparecerá la confirmación y además de eso en nuestro directorio automáticamente se creará una carpeta de nombre **node_modules** aquí es donde se almacenan todas las dependencias y librerías que estaremos instalando mediante comandos **npm** y que más adelante estaremos utilizando en nuestro proyecto. Por ejemplo dentro de esta carpeta encontraremos el paquete de Electron que acabamos de instalar.

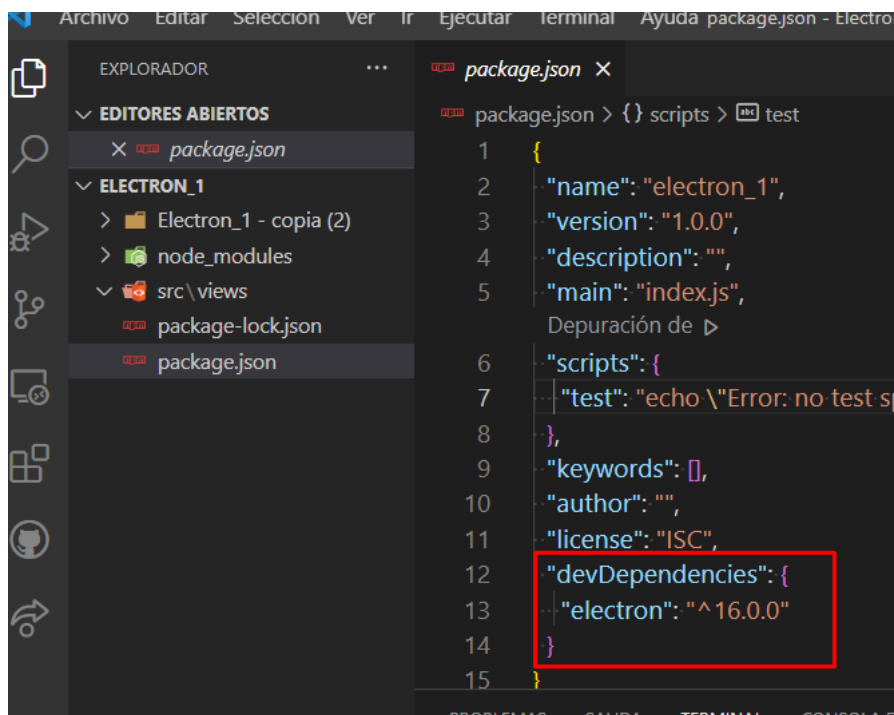


Figura 16: CONFIRMACIÓN INSTALACIÓN

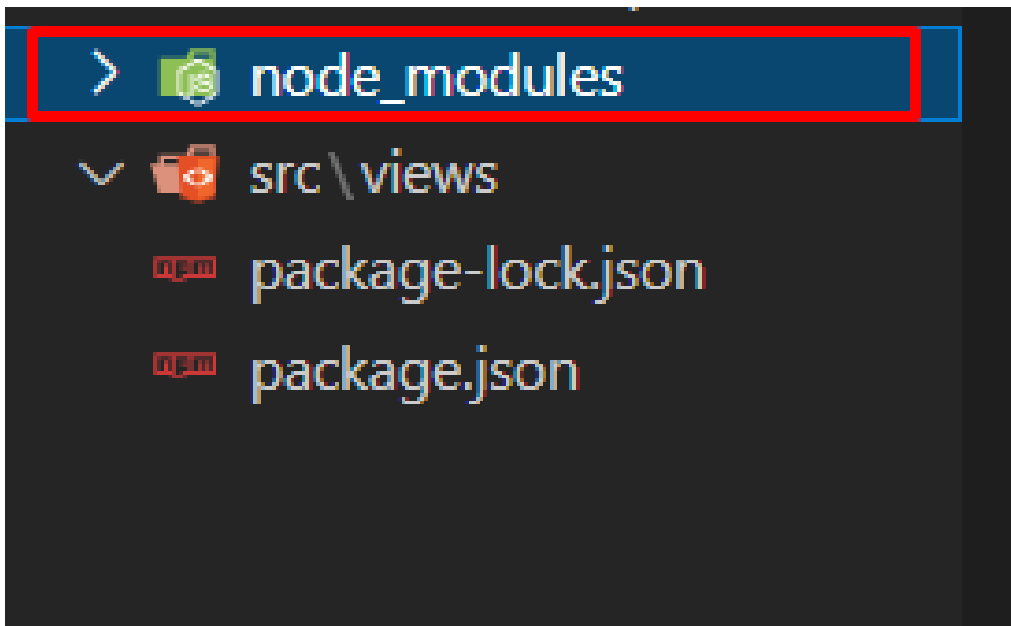


Figura 17: CARPETA NODE_ MODULE

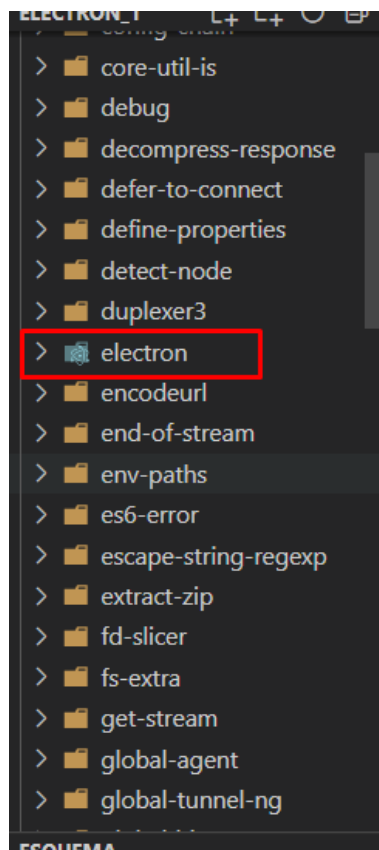
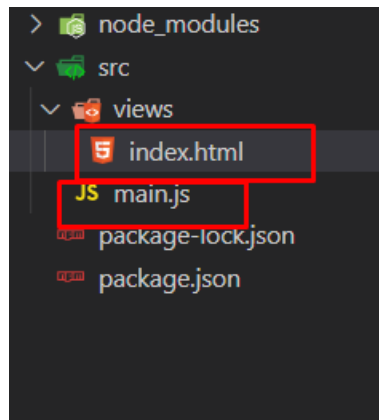


Figura 18: PAQUETE DE ELECTRON

8. Crearemos dos archivos vacíos. Uno de nombre: **main.js**, dentro de **src** y otro de nombre: **index.html** dentro de la carpeta **views**.



9. Ahora realizaremos algunas modificaciones en el archivo **package.json**

```
1  {
2    "name": "electron_1",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "start": "electron ."
8    },
9    "keywords": [],
10   "author": "",
11   "license": "ISC",
12   "devDependencies": {
13     "electron": "^16.0.0"
14   }
15 }
```

a)

ENLACES WEB

APP: Google Play.

URL1: <https://play.google.com/store/apps/details?id=com.teacapps.barcodescanner>

Canal de YouTube: John Ortiz Ordoñez. Curso de Electron

URL2:

<https://youtube.com/playlist?list=PL2PZw96yQChzi-1Lw6rqqAPRkNXSOeqtr>

Canal de YouTube: Códigofacilito. Ejemplo Node.js

URL3: <https://youtu.be/wd8zf3D0jic>

Página Oficial: Visual Studio Code.

URL4: <https://code.visualstudio.com/>

Página Oficial: Node.js.

URL5: <https://nodejs.org/es/>

Página Oficial: Electron.js.

URL6: <https://www.electronjs.org/>

Tutorial Educativo Electron.

URL7: <https://www.electronjs.org/docs/latest/tutorial/quick-start>

Página Oficial de Git

URL8: <https://git-scm.com/>

Videotutorial Instalación de Git

URL9: <https://youtu.be/h9ZH2wFpSUc>

Tutorial Web Instalación de Git

URL10: <https://www.mclibre.org/consultar/informatica/lecciones/git-instalacion.html>

Página Oficial de GitHub

URL11: <https://github.com/>

Tutorial web de registro GitHub

URL12: <https://git-scm.com/book/es/v2/GitHub-Creaci%C3%B3n-y-configuraci%C3%B3n-de-la-cuenta>

Referencias

- Chromium*. (2021, Nov). Wikimedia Foundation. Descargado 2021-11-15, de [https://es.wikipedia.org/wiki/Chromium_\(navegador\)](https://es.wikipedia.org/wiki/Chromium_(navegador))
- Electron*. (s.f.). Descargado 2021-11-18, de <https://www.electronjs.org/>
- Gabriela, M. (2021). *¿qué es y cuál es su función en internet?* Descargado 2021-10-08, de <https://rockcontent.com/es/blog/framework/>
- Introducción a GIT | Tutorial GIT / GITHUB*. (s.f.). Descargado 2021-11-26, de <https://bluuweb.github.io/tutorial-github/01-fundamentos/#%C2%BFque-es-git>
- Muradas, Y. (2021, Sep). *Qué es npm*. OpenWebinars. Descargado 2021-11-17, de <https://openwebinars.net/blog/que-es-node-package-manager/>
- Simões, C. (2021, Jul). *¿qué es node.js?* ITDO Desarrollo web y APPs Barcelona. Descargado 2021-11-15, de <https://www.itdo.com/blog/que-es-node-js-y-para-que-sirve/>
- Strien, D. v. (2017, abril). Introducción al control de versiones con GitHub Desktop (A. R. Castro, Traduc.). *Programming Historian en español*(1). Descargado 2021-11-26, de <https://programminghistorian.org/es/lecciones/retirada/introduccion-control-versiones-github-desktop> doi: 10.46430/phes0015