

## UD 13.CORDOVA Y ELECTRON

**Desarrollo web en entorno cliente  
CFGS DAW**

Sergio García Barea  
[sergio.garcia@ceedcv.es](mailto:sergio.garcia@ceedcv.es)

2019/2020

Versión:191022.1000


## Licencia



**Reconocimiento - NoComercial - CompartirIgual (by-nc-sa):** No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

## ÍNDICE DE CONTENIDO

<b>1. Introducción.....</b>	<b>3</b>
<b>2. Instalación de NodeJS, Cordova y Electron.....</b>	<b>3</b>
<b>3. Docker + Kitematic.....</b>	<b>4</b>
<b>4. Usando Cordova.....</b>	<b>5</b>
4.1 Pasar nuestra página de la máquina real a la máquina Cordova.....	5
4.2 Preparamos el proyecto en el la máquina "Cordova".....	5
4.3 Ponemos nuestra Web en su directorio adecuado.....	6
4.4 Creamos el APK.....	6
4.5 Traemos el APK a la máquina real.....	6
<b>5. Empezando con Electron.....</b>	<b>6</b>
5.1 Primeros pasos con Electron.....	7
5.2 Conectando al NoVNC de Docker.....	7
5.3 Descargando y ejecutando ejemplo de Electron.....	7
5.4 Entendiendo como funciona Electron.....	8
<b>6. Usando Electron y JQuery.....</b>	<b>8</b>
<b>7. Empaquetar nuestra aplicación Electron para distribuirla.....</b>	<b>9</b>
7.1 Empaquetando la aplicación en un directorio.....	9
7.2 Copiar el paquete generado a la máquina real.....	10
<b>8. Acceso a funciones especiales en Cordova y Electron.....</b>	<b>10</b>
<b>9. Enlaces interesantes sobre Cordova Y Electron.....</b>	<b>11</b>

## UD13. CORDOVA Y ELECTRON

### 1. INTRODUCCIÓN

En esta unidad vamos a ver dos herramientas para crear aplicaciones multiplataforma.

En primer lugar, hablaremos de Cordova, que permitirá que vuestras aplicaciones HTML/Javascript se puedan meter en paquetes APK y verse como aplicaciones móviles, además de permitir el acceso a funciones del móvil (cámara, posicionamiento, etc.).

En segundo lugar, hablaremos de Electron, que permitirá que vuestras aplicaciones HTML/Javascript se puedan usar como aplicaciones de escritorio en Windows, Linux y Mac mediante el framework Electron, permitiendo incluso a funciones de acceso al sistema que no se permiten habitualmente en Javascript Web (por ejemplo, el acceso a ficheros).

Para que conozcáis la potencia de Electron, deciros que existen gran cantidad de aplicaciones que han sido creadas usándolo. Podéis ver la lista en <https://electronjs.org/apps>

Entre ellas destacan algunas como el editor Atom, Kitematic o Skype.

### 2. INSTALACIÓN DE NODEJS, CORDOVA Y ELECTRON

En este apartado comentamos como instalar NodeJS ya que Electron depende de él. No es necesario hacer esta instalación en entornos reales, ya que como veremos más adelante mediante Docker podremos descargar entornos ya preparados, pero aun así aquí tenéis una claves para su instalación.

Podemos instalar en cualquier máquina NodeJS <https://nodejs.org/es/> para programar en Javascript para escritorio/servidor. En su web tenéis las instrucciones para instalar NodeJS. Al instalar NodeJS también se nos instalará su potente gestor de paquetes NPM.

Una alternativa para descargar NodeJS (y otros entornos de desarrollo) y gestionar sus versiones de manera óptima (sobretudo en Windows) es el software de instalación de herramientas relacionadas con el desarrollo "Anaconda" <https://www.anaconda.com/download/>

Una vez instalado Nodejs, aquí tenemos las instrucciones para instalar Cordova <https://cordova.apache.org/> y Electron <https://electronjs.org/>

- `npm install -g cordova`
    - Cordova para generar APK requiere que se instale el kit de desarrollo Android.
- `npm install -g electron`

Donde -g indica que instalamos dichos paquetes para todo el sistema (si lo ponemos sin -g, nos lo instala únicamente para el proyecto que esté en el directorio actual).

También podemos instalar otras herramientas como por ejemplo la biblioteca jQuery con el comando

```
npm install -g jquery
```

Pero si por comodidad no queréis instalar todo el software de “NodeJS + Cordova + Android” o “NodeJS + Electron”, recomiendo usar Docker + Kitematic.

### 3. DOCKER + KITEMATIC

Para no tener que instalar todo el software de “Cordova” y “Electron” y hacernos la vida más fácil, recomiendo usar Docker + Kitematic.

Docker es un Software de virtualización y Kitematic es su interfaz gráfica. Es muy sencillo y potente.

Para más información de que es Docker <https://es.wikipedia.org/wiki/>

Aquí algunos tutoriales de como instalar Docker + Kitematic en Windows, Ubuntu Linux y Mac Os.

<https://videotutoriales.com/illasaron/2017/01/13/02-curso-docker-instalar-docker-toolbox-en-windows/>

[https://docs.docker.com/toolbox/toolbox\\_install\\_windows/](https://docs.docker.com/toolbox/toolbox_install_windows/)

<https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/#install-using-the-repository>

[https://docs.docker.com/toolbox/toolbox\\_install\\_mac/#step-2-install-docker-toolbox](https://docs.docker.com/toolbox/toolbox_install_mac/#step-2-install-docker-toolbox)

Nota: si no queremos necesitar ser superusuario para arrancar Docker en Linux, debemos seguir estos pasos <https://docs.docker.com/install/linux/linux-postinstall/>

Una vez instalado Docker + Kitematic, arrancamos este último (“gksudo kitematic” en Linux). En Kitematic con la opción “new” nos deja buscar máquinas Docker para instalar. Estas las busca automáticamente de la plataforma Docker Hub <https://hub.docker.com/>

Las máquinas que recomendamos instalar son:

- Para Cordova:
  - Instalaremos la máquina <https://hub.docker.com/r/beevelop/cordova/>
  - La forma más sencilla de encontrarla en Kitematic es buscar "cordova" e instalamos la primera máquina "beevelop/cordova" que nos aparece.
- Para Electron
  - Instalaremos la máquina <https://hub.docker.com/r/sergarb1/ceedcv-taller-electron-nodejs/>

Cualquier duda de uso de Docker + Kitematic podéis comentarla en el foro.

## 4. USANDO CORDOVA

Supongamos tenemos una aplicación Web que queremos empaquetar en una APK para Android. Dicha aplicación puede ser una página o varias, incluir recursos como ficheros y bibliotecas JS, imágenes, etc. pero la inicial debe llamarse index.html).

Para facilitar el tutorial, suponemos que todo el contenido lo tenemos comprimido en el fichero "nuestraPagina.zip".

### 4.1 Pasar nuestra página de la máquina real a la máquina Cordova

Para pasar ese fichero de nuestra maquina real a la máquina "cordova" de Docker, ejecutamos en una terminal Linux (o en Windows en "Docker terminal") el comando:

```
"sudo docker cp /nuestraPagina.zip cordova:/tmp/"
```

Donde "cordova:" indica el nombre o identificador del contenedor de Docker (Si fuera un Docker que se llamara "pepito", pondríamos "pepito:").

Esta orden copiaría "/nuestraPagina.zip" de la maquina real al directorio /tmp del contenedor Docker "cordova".

### 4.2 Preparamos el proyecto en el la máquina "Cordova"

Una vez esto, vamos a una Shell en la maquina "cordova" (para ello en Kitematic seleccionamos la maquina y hacemos click en "Exec").

Por defecto el tipo de Shell es "/bin/sh", si queréis usar el "/bin/bash" simplemente teclear "/bin/bash" y os cambiara la Shell.

Una vez en la Shell de la maquina "cordova" escribimos:

```
"cordova create MiProyecto com.pruebas.miproyecto MiProyecto".
```

Donde el primer parámetro es el nombre de la aplicación, el segundo es la ruta Android de la app y el tercero es el directorio a crear con el proyecto.

Una vez hecho eso, tendremos un directorio "MiProyecto".

Entramos a esa carpeta "cd MiProyecto" y ahí escribimos "cordova platform add android". Esto sirve para decirle que genere código Android (un APK).

Cordova no se limita a APK, también se puede genera código IOS por ejemplo.

#### 4.3 Ponemos nuestra Web en su directorio adecuado

Una vez hecho esto, en MiProyecto hay una carpeta "www".

Podéis borrar su contenido y meter vuestro código web. Recordad que debe haber una página "index.html".

Si tenéis un zip podéis descomprimirlo con el comando "unzip"

```
"unzip nuestraPagina.zip www"
```

#### 4.4 Creamos el APK

Una vez descomprimido, creamos el APK con la orden

```
"cordova build" para la version "debug" o para sacar la versión de producción  
"cordova build --release"
```

Así se creara el APK. Al finalizar nos indicará donde se aloja el APK. Generalmente (aunque podría cambiar según versión) se encontrará en la carpeta del proyecto "platforms/android/build/".

#### 4.5 Traemos el APK a la máquina real

Para traerla a tu maquina real, volvemos a la consola (o "Docker terminal" en Windows) y usamos de nuevo una orden similar a la vista anteriormente

```
"sudo docker cp cordova:/rutaproyecto/fichero.apk /rutamaquinareal".
```

Con esta orden copiamos el APK de la máquina "cordova" al directorio actual de la maquina real. Con eso, ya tenemos el APK. Solo debemos instalarlo en el móvil/tablet/etc. y ver que funcion.

### 5. EMPEZANDO CON ELECTRON

Buscamos en Kitematic la máquina personalizada con el entorno Electron "<https://hub.docker.com/r/sergarb1/dwec-ceedcv-nodejs-electron-quasar>".

En concreto es esta <https://hub.docker.com/r/sergarb1/ceedcv-taller-electron-nodejs/> Esa máquina tiene todo el software instalado y configurado para poder usar "NodeJS + Electron". Además de todo esto, permite conectarse al escritorio de la máquina via "NoVNC" (herramienta de administración remota cuyo cliente es un navegador Web) y así podemos ver las aplicaciones resultantes en funcionamiento.

Más información sobre NoVNC <https://www.redeszone.net/2015/06/20/novnc-un-cliente-vnc-basado-en-html5-con-seguridad-wss/>

## 5.1 Primeros pasos con Electron

Suponiendo tenemos ya todo el software instalado (ya sea manualmente o porque hemos bajado el Docker solicitado), podemos hacer nuestra primera prueba con el “Hola mundo” de Electron.

## 5.2 Conectando al NoVNC de Docker

Si estamos con el Docker personalizado del CEED, lo más práctico para todo esto es conectarnos al escritorio de la máquina mediante NoVNC. Para ello, debemos observar mediante Kitematic a que puerto se redirige el puerto 6080 (el de NoVNC) y conectarnos a dicho puerto usando un navegador Web.

En mi caso por ejemplo se redirige al 32768 por lo cual en mi navegador escribo localhost:32678 y accedo al escritorio Linux mediante NoVNC (Nota: si usáis Windows Toolbox de Docker, en lugar de localhost será otra IP que también podéis ver en Kitematic).

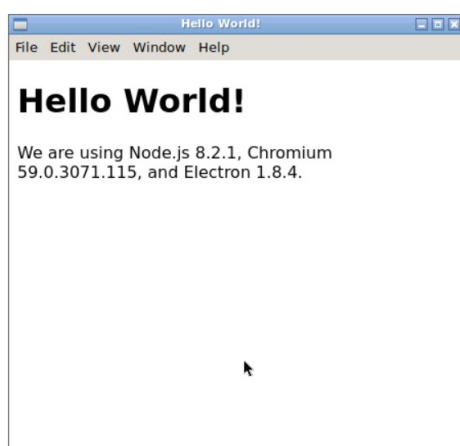
Una vez dentro del Escritorio, para utilizar comandos de consola podemos usar alguna de las existentes, como “Xterm”.

## 5.3 Descargando y ejecutando ejemplo de Electron

Siguiendo el tutorial básico, ejecutamos las siguientes acciones en una consola, pero dentro de un entorno de escritorio (en caso contrario, el “npm start” fallará).

```
# Clona el repositorio con el ejemplo base de Electron
git clone https://github.com/electron/electron-quick-start
# Accede al repositorio clonado
cd electron-quick-start
# Instala dependencias de la aplicación (si las hay)
npm install
# Lanza la aplicacion
npm start
```

Con esto lanzaremos la aplicación, con un resultado similar al de la imagen.



ceedcv  
CENTRE ESPECÍFIC  
D'EDUCACIÓ A DISTÀNCIA DE  
LA COMUNITAT VALENCIANA

```

root@11d3e0ee4928: ~/electron-quick-start
Receiving objects: 100% (273/273), 45.73 KiB | 0 bytes/s, done.
Resolving deltas: 100% (127/127), done.
Checking connectivity... done.
root@11d3e0ee4928:~/electron-quick-start# npm install

> electron@1.8.4 postinstall /home/ubuntu/electron-quick-start/node_modules/electron
> node install.js

Downloading SHASUMS256.txt
[=====] 100.0% of 3.43 kB (3.43 kB/s)
[=====] created a lockFile as package-lock.json. You should commit this file.
added 155 packages in 206,247s
root@11d3e0ee4928:~/electron-quick-start# npm start

> electron-quick-start@1.0.0 start /home/ubuntu/electron-quick-start
> electron .

process 224: D-Bus library appears to be incorrectly set up: failed to read each
line uid: UID file '/etc/machine-id' should contain a hex string of length 32,
not length 0, with no other text
See the manual page for dbus-uuidgen to correct this issue.

```

## 5.4 Entendiendo como funciona Electron

En este primer ejemplo que hemos probado hay 3 ficheros importantes:

- **package.json:** Contiene información de la aplicación (detalles, dependencias) y apunta al fichero principal.
- **main.js:** Fichero que arranca la aplicación y que crea el navegador dentro de una ventana para que se pueda ver nuestra aplicación HTML/Javascript. Es el principal proceso de la aplicación.
- **index.html:** la Web que mostramos en nuestra aplicación. Este fichero lo sustituiríamos por el index de nuestra aplicación.

Para saber más, aquí la guía de inicio de Electron <https://electronjs.org/docs/tutorial/quick-start>

## 6. USANDO ELECTRON Y JQUERY

Si intentamos usar JQuery con Electron tendremos problemas. Deberemos adaptar nuestro programa con una serie de sencillos pasos.

Tomamos la información de como adaptarlo de la segunda y tercera soluciones propuestas en

<https://ourcodeworld.com/articles/read/202/how-to-include-and-use-jquery-in-electron-framework>

### Solución A

1) Instalamos jQuery para NodeJs. Podemos hacerlo con el siguiente comandos

```
npm install jquery -g
```

2) Tras ello, sustituimos la línea donde incluimos la librería de JQuery por esta

```
<script>>window.$ = window.jQuery = require('jquery');</script>
```

Con estos cambios nuestra aplicación deberá funcionar correctamente.



## Solución B

En esta solución simplemente sustituimos la línea donde se incluye JQuery por esta, donde "rutaJquery.js" es la ruta local donde esta la biblioteca.

```
<script>window.$ = window.jQuery = require('./rutaJquery.js');</script>
```

## Probando las soluciones

Podemos probar esta mediante un ejemplo de aplicación en JQuery que requiere adaptación par funcionar en Electron. La tenéis disponible en <http://hispabyte.net/ejemploPaint.zip>

Podemos bajarla a la máquina Linux con el comando

```
wget http://hispabyte.net/ejemploPaint.zip
```

Con las adaptaciones propuestas debería funcionar sin problemas.

## 7. EMPAQUETAR NUESTRA APLICACIÓN ELECTRON PARA DISTRIBUIRLA

Vamos a explicar como generar un directorio que contenga todo lo necesario para distribuir la aplicación.

✎ Debemos generar un paquete por cada Sistema/Arquitectura que queremos que soporte y distribuir dichos directorios completos.

Vamos a presentar unos ejemplos de creación de paquetes para distribuir. Para estos ejemplos nos basamos en la información sobre "electron-packager" obtenida en

<https://www.christianengvall.se/electron-packager-tutorial/>

### 7.1 Empaquetando la aplicación en un directorio

Podemos instalar la herramienta "electron-packager" con el comando

```
npm install electron-packager -g
```

Una vez instalado, pongo aquí un ejemplo para generar un paquete desde Linux usando como destino una plataforma Linux de 64 bits.

```
electron-packager . miAPP --overwrite --platform=linux --arch=x64
```

Sobre este comando indicar:

- El "." Inicial indica donde está el proyecto Electron (en este ejemplo, suponemos estamos en dicho directorio).
- miAPP es el nombre de la aplicación. Se generará un directorio formado por su nombre seguido de la plataforma y arquitectura. En dicho directorio es donde se guardara la aplicación con todo lo necesario para funcionar, incluyendo un ejecutable llamado "miAPP".

- `--overwrite` indica que si ya había un proyecto en el destino, lo sobrescriba.
- `--platform` indica la plataforma. En este caso Linux.
- `--arch` indica la arquitectura, en este caso x64

Otro ejemplo con el comando equivalente para crear un paquete para Windows de 32 bits.

```
electron-packager . miAPP --overwrite --platform=win32 --arch=ia32
```

NOTA: para crear aplicaciones Windows desde Linux, es obligatorio tener instalado Wine.

## 7.2 Copiar el paquete generado a la máquina real

Al finalizar el empaquetado la forma más sencilla de pasarla de la maquina Docker a la maquina real, es empaquetando la carpeta en un zip en la maquina docker y copiando el fichero a la maquina real con comandos Docker.

Un posible comando para comprimir en zip


```
tar -czvf miAPP.tar.gz miAPP-linux-x64/
```

Para traerla a tu maquina real, volvemos a la consola (o "Docker terminal" en Windows) y usamos de nuevo una orden similar a la vista anteriormente

```
sudo docker cp ceedcv-taller-electron-nodejs:/rutazip/miCarperta.tar.gz ./
```

Con esta orden copiamos el ZIP de la máquina "ceedcv-taller-electron-nodejs" al directorio actual de la maquina real.

Con eso, ya tenemos la aplicación y podemos distribuirla al sistema que queramos.

 Una vez distribuida la aplicación y descomprimida la carpeta en la máquina destino, el usuario simplemente deberá ejecutar un ejecutable con el nombre de la aplicación. **Ejemplo:** `./miAPP` en Linux, `miApp.exe` en Windows

## 8. ACCESO A FUNCIONES ESPECIALES EN CORDOVA Y ELECTRON

Tanto Cordova como Electron tienen acceso a funciones especiales de los dispositivos. Por ejemplo Cordova permite en móviles acceder a la camara, al GPS, etc. y Electron permite acceder a funciones del sistema operativo como leer ficheros locales, escribirlos, etc.

Dichas funcionalidades se mostrarán en el curso como ejemplos prácticos comentados, siendo esta unidad únicamente una base para conocer Cordova y Electron, sin profundizar en ellos.

## 9. ENLACES INTERESANTES SOBRE CORDOVA Y ELECTRON

Aquí algunos enlaces interesantes sobre Cordova y Electron:

### **Cordova**

Instalar Cordova en Ubuntu.

<https://rolandocaldas.com/android/instalar-cordova-en-ubuntu>

Ejemplos de Cordova

<https://github.com/cfjedimaster/Cordova-Examples>

### **Electron**

Ejemplos de aplicaciones Electron y recurso (videos, tutoriales) creadas por la comunidad

<https://electronjs.org/community#boilerplates>

Tutorial inicio rápido

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

Aplicación hecha con Electron que... te enseña a usar Electron :)

<https://github.com/electron/electron-api-demos>