**MANUAL**

| **INFORMACIÓN GENERAL** | |
| --- | --- |
| **Fecha** |  |
| **Nombre de la asignatura** | Electiva de Profundización I (Electrón) |
| **Docente** | Oscar Eduardo Ortiz Pinzón |
| **Horario** | Miércoles 6:00 pm – 8:00 pm |
| **Grupo** | II |

| TALLER DE CLASE |
| --- |
| Tener instalado en sus equipos los siguientes componentes.   1. Visual Studio Code (o Atom si les parece, https://code.visualstudio.com) 2. Tener Instalado NodeJS (Version LTS https://nodejs.org/es/) 3. Para validar la opción debe tener una consola de Windows (Tecla Windows + R) 4. En dicha consola debe escribir lo siguiente: node --version 5. Deberá aparecer la referencia que hemos descargado 6. Opcional tener instalado los distintos plug ins 7. Auto Rename Tag 8. Bracket Pair Colorizer 9. ESLint 10. Live Server 11. Prettier - Code formatter 12. Project Manager 13. REST Client 14. Crear una carpeta (ustedes deciden donde crean la carpeta, el único consejo es que sea de fácil recordación, que sea una ruta de máximo 256 caracteres). 15. En dicha carpeta abierta desde Visual Studio Code debe hacer lo siguiente: 16. Debe entrar en la consola (Ver o View – Consola o Terminal (Ctrl + ñ)) 17. Deberá aparecer lo siguiente:     Y allí vamos a adelantar el trabajo.  Vamos a la carpeta que hemos creado para tal fin.   1. Después escribimos lo siguiente: npm init      1. Nos aparecerá luego nuestro primer referencia con NPM lo siguiente     Determinar en cada caso o cada estudiante los datos pedidos.   1. Se nos ha creado un archivo del tipo: package.json 2. Entramos en el tipo package.json que se nos despliega      1. Remplazar en este punto donde aparece   "test": "echo \"Error: no test specified\" && exit 1"  Por esto:  "start": "electron ."  Darle Ctrl + S (o guardar)   1. Luego después de esto darle install a electron   Bien sea local (npm install --save-dev electron) o  Bien sea Global (npm install --g electron)       1. Luego de esto podríamos hacer lo siguiente: crear dos tipos de archivos, el primero de tipo javascript y el otro podríamos crear el código en html.   El primero debe tener la misma extensión que nos aparece en “main”: “index.js”  Y el segundo sería el mismo archivo que hace referencia la aplicación “index.html”     1. En el archivo definido como index.js copiamos y pegamos lo siguiente, entendiendo como cada línea de código nos da el ejemplo  |  | | --- | | const { app, BrowserWindow } = require('electron')  function createWindow () {  // Crea la ventana del navegador.  const win = new BrowserWindow({  width: 800,  height: 600,  webPreferences: {  nodeIntegration: true  }  })  // y carga el index.html de la aplicación.  win.loadFile('index.html')  // Abre las herramientas de desarrollo (DevTools).  win.webContents.openDevTools()  }  // Este método se llamará cuando Electron haya finalizado  // la inicialización y esté preparado para crear la ventana del navegador.  // Algunas APIs pueden solamente ser usadas despues de que este evento ocurra.  app.whenReady().then(createWindow)  // Quit when all windows are closed, except on macOS. There, it's common  // for applications and their menu bar to stay active until the user quits  // explicitly with Cmd + Q.  app.on('window-all-closed', () => {  if (process.platform !== 'darwin') {  app.quit()  }  })  app.on('activate', () => {  // On macOS it's common to re-create a window in the app when the  // dock icon is clicked and there are no other windows open.  if (BrowserWindow.getAllWindows().length === 0) {  createWindow()  }  })  // En este archivo puede incluir el resto del código del proceso principal específico  // de su aplicación. Tu también puedes ponerlos en archivos separados y requerirlos aquí. |  1. Luego de esto, debemos copiar Y pegar conociendo como sería el archivo que necesitamos hacer.  |  | | --- | | <!DOCTYPE html>  <html>  <head>  <meta charset="UTF-8">  <title>Hello World!</title>  <!-- https://electronjs.org/docs/tutorial/security#csp-meta-tag -->  <meta http-equiv="Content-Security-Policy" content="script-src 'self' 'unsafe-inline';" />  </head>  <body>  <h1>Oscardo</h1>  We are using node <script>document.write(process.versions.node)</script>,  Chrome <script>document.write(process.versions.chrome)</script>,  and Electron <script>document.write(process.versions.electron)</script>.  </body>  </html> |   Vamos a entender todo lo que hemos pegado:   1. const { app, BrowserWindow } = require('electron') //determinamos que elementos del sistema nosotros podemos abstraer. 2. Creamos una ventana   function createWindow () {    // Crea la ventana del navegador.    const win = new BrowserWindow({      width: 800,      height: 600,      webPreferences: {        nodeIntegration: true      }    })  Determinamos el tamaño que deseamos asociar  Y las preferencias, si nosotros deseamos utilizar la integración por Node o si nosotros deseamos tener otro gestor, en este momento dejamos true.   1. El tipo de archivo (html) y la integración con el openDevTools (opcional)     // y carga el index.html de la aplicación.    win.loadFile('index.html')    // Abre las herramientas de desarrollo (DevTools).    win.webContents.openDevTools()  // Este método se llamará cuando Electron haya finalizado  // la inicialización y esté preparado para crear la ventana del navegador.  // Algunas APIs pueden solamente ser usadas despues de que este evento ocurra.  app.whenReady().then(createWindow)  // Quit when all windows are closed, except on macOS. There, it's common  // for applications and their menu bar to stay active until the user quits  // explicitly with Cmd + Q.  app.on('window-all-closed', () => {    if (process.platform !== 'darwin') {      app.quit()    }  })  app.on('activate', () => {    // On macOS it's common to re-create a window in the app when the    // dock icon is clicked and there are no other windows open.    if (BrowserWindow.getAllWindows().length === 0) {      createWindow()    }  })  // En este archivo puede incluir el resto del código del proceso principal específico  // de su aplicación. Tu también puedes ponerlos en archivos separados y requerirlos aquí.   1. Luego de esto, debemos vemos el control html      1. Después de esto debemos darle a la aplicación: npm start  |  |  | | --- | --- | |  |  |  1. Estado inicial del Taller.   Ejecuta el primer archivo de la carpeta llamada: Shell |
|  |
| Hacer un Github (<https://github.com/CURepublicana/ElectivaProfundizacion1>) |