



PROCESO DE GESTIÓN DE FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO MATERIAL DE APOYO – ACTIVIDADES

MATERIAL DE APOYO MANIPULACIÓN DOM

## ¿Qué es el DOM?

Las siglas **DOM** significan **Document Object Model**, o lo que es lo mismo, la estructura de un documento HTML. Una página HTML está formada por múltiples etiquetas HTML, anidadas una dentro de otra, formando un árbol de etiquetas relacionadas entre sí, que se denomina **árbol DOM** (o simplemente DOM).

Ejemplo: Tenemos el siguiente código html

```
MANEJODOM > <> ejer2.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6  </head>
7  <body>
8      <div id="principal">
9          <div class="cabeza">
10             <h2>EJEMPLO HTML MANEJO DOM</h2>
11          </div>
12          <div class="menu">
13             <button class="boton">UNO</button>
14             <button class="boton" id="btnDos">DOS</button>
15             <button class="boton" disabled>TRES</button>
16          </div>
17          <div class="contenido">
18             <h3>LISTA EJEMPLO</h3>
19             <ul id="ul-ejemplo">
20                 <li>Uno</li>
21                 <li>Dos</li>
22                 <li>Tres</li>
23             </ul>
24             <input type="text" name="txtNombre" id="txtNombre" placeholder="Ingrese Nombre">
25             <input type="text" name="txtNombre" id="txtApellido" placeholder="Ingrese Apellido">
26             <button id="btnMostrar">Mostrar</button>
27          </div>
28          <div class="piePagina">
29             <p id="textoPiePagina">ADSO - CAUCA - 2025</p>
30          </div>
31      </div>
32      <script src="ejer2.js"></script>
33  </body>
34  </html>
```



Cuya visualización en el navegador es:

# EJEMPLO HTML MANEJO DOM

UNO

DOS

TRES

## LISTA EJEMPLO

- Uno
- Dos
- Tres

Mostrar

ADSO - CAUCA - 2025

Desde el navegador podemos visualizar nuestro árbol con la herramienta inspeccionar.

The screenshot shows a web browser with the URL `127.0.0.1:5500/MANEJODOM/ejer2.html`. The page content is identical to the one above. On the right, the 'Elementos' (Elements) panel is open, displaying the DOM tree. The tree structure is as follows:

- `<DOCTYPE html>`
- `<html lang="en">`
- `<head>`
  - `<meta charset="UTF-8">`
  - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
- `</head>`
- `<body>`
  - `<div id="principal">`
    - `<div class="cabecera">`
      - `<h2>EJEMPLO HTML MANEJO DOM</h2>`
      - `</div>`
      - `<div class="menu">`
        - `<button class="boton">UNO</button>`
        - `<button class="boton" id="btnDos">DOS</button>`
        - `<button class="boton" disabled="true">TRES</button>`
      - `</div>`
      - `<div class="contenido">`
        - `<h3>LISTA EJEMPLO</h3>`
        - `<ul id="ejemplo">`
          - `<li></li>`
          - `<li></li>`
          - `<li></li>`
        - `</ul>`
        - `<input type="text" name="txtNombre" id="txtNombre" placeholder="Ingrese Nombre">`
        - `<input type="text" name="txtApellido" id="txtApellido" placeholder="Ingrese Apellido">`
        - `<button id="btnMostrar">Mostrar</button>`
      - `</div>`
      - `<div class="piePagina"></div>`
      - `</div>`
      - `<script src="ejer2.js"></script>`
      - `<!-- Code injected by live-server -->`
      - `</script>`
    - `</body>`
    - `</html>`



Desde **Javascript** podemos acceder a los elementos y modificarlos si es necesario.

### ¿Cómo seleccionar elementos desde Javascript?

- **getElementById(id)**: Acceder al elemento por el **id**
- **getElementsByClassName(class)**: Acceder a los elementos de acuerdo a una **clase**
- **getElementsByName(value)**: Acceder a los elementos de acuerdo con el nombre de la propiedad **name**
- **getElementsByTagName(tag)**: Acceder a los elementos de acuerdo con el nombre de la **etiqueta** html.
- **querySelector(sel)**: Busca el primer elemento que coincide con el selector **sel**
- **querySelectorAll(sel)**: Busca todos los elementos que coinciden con el selector **sel**

### ¿Cómo seleccionar elementos desde Javascript?

Ejemplos de acuerdo al código html presentado al inicio del documento.

```
<ul id="ul-ejemplo">
  <li>Uno</li>
  <li>Dos</li>
  <li>Tres</li>
</ul>
```

El código Javascript para obtener el elemento con **id=ul-ejemplo**

```
const listaUL = document.getElementById("ul-ejemplo")
console.log("Elemento consulta por su ID")
console.log(listaUL)
```



## Resultado en consola

```
Elemento consulta por su ID
▼ <ul id="ul-ejemplo">
  ▼ <li>
    ::marker
    "Uno"
  </li>
  ▼ <li>
    ::marker
    "Dos"
  </li>
  ▼ <li>
    ::marker
    "Tres"
  </li>
</ul>
```

## Ejemplo mediante getElementByClassName(class):

```
<div class="menu">
  <button class="boton">UNO</button>
  <BUTTON class="boton" id="btnDos">DOS</BUTTON>
  <BUTTON class="boton" disabled>TRES</BUTTON>
</div>
```

Código Javascript para acceder a los elementos que tengan la clase llama botón

```
const elementosPorClase = document.getElementsByClassName("boton")
console.log("Elementos por clase llamada boton")
console.log(elementosPorClase)
```

**Resultado:** El resultado es un objeto de tipo HTMLCollection con 3 elementos que tienen esa clase.

```
Elementos por clase llamada boton
▼ HTMLCollection(3) [button.boton, button#btnDos.boton, button.boton, btnDos: button#btnDos.boton] ⓘ
  ▶ 0: button.boton
  ▶ 1: button#btnDos.boton
  ▶ 2: button.boton
  ▶ btnDos: button#btnDos.boton
  length: 3
  ▶ [[Prototype]]: HTMLCollection
```



### Ejemplo mediante `getElementsByName(value)`:

```
<input type="text" name="txtNombre" id="txtNombre" placeholder="Ingrese Nombre">  
<input type="text" name="txtNombre" id="txtApellido" placeholder="Ingrese Apellido">
```

Código javascript para acceder a los elementos que tengan la propiedad **name** con el mismo valor.

```
const elementosPorValorPropiedadName= document.getElementsByName("txtNombre")  
console.log("Elementos cuya propiedad name es txtNombre")  
console.log(elementosPorValorPropiedadName)
```

### Resultado:

```
Elementos cuya propiedad name es txtNombre  
▼ NodeList(2) [input#txtNombre, input#txtApellido] ⓘ  
  ▶ 0: input#txtNombre  
  ▶ 1: input#txtApellido  
    length: 2  
  ▶ [[Prototype]]: NodeList
```

### Ejemplo mediante `getElementsByTagName(tag)`:

```
<ul id="ul-ejemplo">  
  <li>Uno</li>  
  <li>Dos</li>  
  <li>Tres</li>  
</ul>
```

Código javascript para obtener los elementos cuya **etiqueta** es **li** en todo el documento

```
const elementosPorNombreEtiqueta = document.getElementsByTagName("li")  
console.log("Elementos cuya etiqueta es li")  
console.log(elementosPorNombreEtiqueta)
```



Resultado:

```
Elementos cuya etiqueta es li
▼ HTMLCollection(3) [li, li, li] ⓘ
  ▶ 0: li
  ▶ 1: li
  ▶ 2: li
    length: 3
  ▶ [[Prototype]]: HTMLCollection
```

Ejemplo mediante `querySelector(sel)`: Ejemplo consulta por ID

```
<div class="piePagina">
  <p id="textoPiePagina">ADSO - CAUCA - 2025</p>
</div>
```

Código javascript para obtener el elemento con id **textoPiePagina** usando **querySelector**

```
const elementoPorIdQuerySelector=document.querySelector("#textoPiePagina")
console.log("Elemento por Id usando query Selector se antepone #")
console.log(elementoPorIdQuerySelector)
```

Resultado

```
Elemento por Id usando query Selector se antepone #
  <p id="textoPiePagina">ADSO - CAUCA - 2025</p>
```

Ejemplo mediante `querySelector(sel)`: Ejemplo consulta por clase

```
<div class="piePagina">
  <p id="textoPiePagina">ADSO - CAUCA - 2025</p>
</div>
```



Código javascript consulta elemento por clase mediante querySelector

```
const elementoPorClaseQuerySelector=document.querySelector(".piePagina")
console.log("Elemento por Clase usando query Selector se antepone .")
console.log(elementoPorClaseQuerySelector)
```

### Resultado

```
Elemento por Clase usando query Selector se antepone .
▼ <div class="piePagina">
  <p id="textoPiePagina">ADSO - CAUCA - 2025</p>
</div>
```

**Ejemplo mediante querySelector(sel):** Ejemplo consulta por etiqueta html

```
const elementoPorEtiquetaQuerySelector=document.querySelector("button")
console.log("Elemento por etiqueta html usando query Selector")
console.log(elementoPorEtiquetaQuerySelector)
```

### Resultado

```
Elemento por etiqueta html usando query Selector
<button class="boton">UNO</button>
```

Como pueden darse cuenta el resultado obtuvo el primer elemento con etiqueta button dentro del documento html. En el documento hay 4 elementos de tipo button.

**Ejemplo mediante querySelectorAll(sel):** Consulta por propiedad type=text

```
const elementosPorAtributo= document.querySelectorAll('[type="text"]')
console.log("Elementos por valor de propiedad usando query Selector")
console.log(elementosPorAtributo)
```



## Resultado:

Elementos por valor de propiedad usando query Selector

```
▼ NodeList(2) [input#txtNombre, input#txtApellido] ⓘ  
  ▶ 0: input#txtNombre  
  ▶ 1: input#txtApellido  
    length: 2  
  ▶ [[Prototype]]: NodeList
```

## Modificar el contenido del Documento

- **textContent:** Devuelve (o cambia) el contenido de texto del elemento.
- **innerHTML:** Devuelve (o cambia) el contenido HTML del interior del elemento.

## Ejemplo texContent:

```
<div class="piePagina">  
  <p id="textoPiePagina">ADSO - CAUCA - 2025</p>  
</div>
```

Código javascript para obtener el contenido de lo que tenga el elemento con id=textoPiePagina

```
const elementoPorIdQuerySelector=document.querySelector("#textoPiePagina")  
console.log("Contenido del elemento con id=textoPiePagina: ")  
console.log(elementoPorIdQuerySelector.textContent)
```

## Resultado

```
Contenido del elemento con id=textoPiePagina:  
ADSO - CAUCA - 2025
```





Ahora vamos a modificar dicho contenido por: **Análisis y Desarrollo de Software – CAUCA – 2025**

Código para cambiarlo

```
elementoPorIdQuerySelector.textContent="Análisis y Desarrollo de Software - CAUCA - 2025"
```

Resultado en el explorador

## EJEMPLO HTML MANEJO DOM

### LISTA EJEMPLO

- Uno
- Dos
- Tres

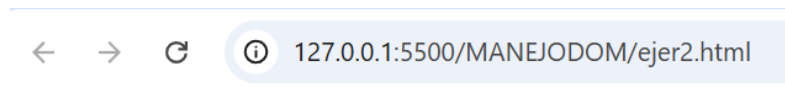
Análisis y Desarrollo de Software - CAUCA - 2025

### Ejemplo usando innerHTML

Ahora vamos agregar en el texto del pie de página mediante código html así:

```
elementoPorIdQuerySelector.innerHTML="<p>Análisis y Desarrollo de Software <b>(ADSO)</b> - CAUCA - 2025 <br>"  
+ "Centro de Teleinformática y Producción industrial <b>CTPI</B> "
```

Resultado en el explorador



## EJEMPLO HTML MANEJO DOM

### LISTA EJEMPLO

- Uno
- Dos
- Tres

Análisis y Desarrollo de Software **(ADSO)** - CAUCA - 2025  
Centro de Teleinformática y Producción industrial **CTPI**



Ejemplo Modificar un elemento cambiando el estilo.

```
const botonUno = document.querySelector("button")
botonUno.style.color="red"
botonUno.style.backgroundColor="yellow"
```

Resultado en el explorador

← → ↻ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

## EJEMPLO HTML MANEJO DOM

**UNO** DOS TRES

### LISTA EJEMPLO

- Uno
- Dos
- Tres

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI

Ejemplo cambiar el texto Uno de la lista Ejemplo por el valor de First

```
const liUno = document.querySelector("li")
liUno.textContent="First"
```

Resultado

← → ↻ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

## EJEMPLO HTML MANEJO DOM

**UNO** DOS TRES

### LISTA EJEMPLO

- First
- Dos
- Tres

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI



## Crear Elementos HTML

- **createElement(tag, options):** Crea y devuelve el elemento HTML definido por el tag.
- **createComment(text):** Crea y devuelve un nodo de comentarios HTML `<!-- text -->`.
- **createTextNode(text):** Crea y devuelve un nodo HTML con el texto text.

### Ejemplo createElement(tag, options)

```
<ul id="ul-ejemplo">
  <li>Uno</li>
  <li>Dos</li>
  <li>Tres</li>
</ul>
```

Vamos a crear elemento de tipo **li** para agregarlo a la lista **ul** con **id="ul-ejemplo"**

```
//createElement

const elementosUl= document.querySelector("#ul-ejemplo")
const liNuevo = document.createElement("li")
liNuevo.textContent="Cuatro"
//agregarlo al nodo que la agrupa
elementosUl.appendChild(liNuevo)
```

En el código anterior vemos el método **nodoPadre.appendChild(elemento)** que nos permite agregar un elemento a un nodo padre.



## Resultado

← → ↻ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

## EJEMPLO HTML MANEJO DOM

**UNO** DOS TRES

### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI

### Ejemplo uso de `createComment(text)`

El método crea un comentario dentro del código html permitiendo documentar el código de acuerdo a las necesidades.

```
const comentario = document.createComment("Lista de Elementos");
elementosUl.appendChild(comentario)
```

El resultado se visualiza dentro del documento html

```
<ul id="ul-ejemplo">
  <li>...</li>
  <li>...</li>
  <li>...</li>
  <li>...</li>
  <!--Lista de Elementos-->
</ul>
```



Ejemplo uso `createTextNode(text)`:


```
const divMenu = document.querySelector(".cabeza")
const texto = document.createTextNode("Menú de Opciones")
divMenu.append(texto)
```

Resultado

← → ↻ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

---

## EJEMPLO HTML MANEJO DOM

Menú de Opciones 

**UNO** DOS TRES

### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Análisis y Desarrollo de Software (**ADSO**) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial **CTPI**



## Acceso a las clases o estilos de CSS

- **.className:** Acceso directo al valor del atributo class. También asignable
- **.classList:** Array de clases CSS. Contiene métodos y propiedades de ayuda
- **.style:** Objeto con las propiedades CSS asignadas en ese elemento.

### Ejemplo uso de className

Vamos a crear una clase en un archivo de tipo css así:

```
MANEJODOM > # ejer2.css > .cajaTexto
1  ✓ .cajaTexto{
2      width: 50%;
3      height: 20px;
4      padding: auto;
5      margin-top: 5px;
6  }
```

El archivo que contiene la clase anterior debe ser llamado desde el documento html

```
<link rel="stylesheet" href="ejer2.css">
```

Ahora vamos asignarle la clase **cajaTexto** a los controles de formulario cuyo **type=text**

```
const elementosPorAtributo= document.querySelectorAll('[type="text"]')
console.log("Elementos por valor de propiedad usando query Selector")
console.log(elementosPorAtributo)
Tabnine | Edit | Test | Explain | Document
elementosPorAtributo.forEach(element => {
    element.className="cajaTexto"
});
```



## Resultado

← → ↻ 127.0.0.1:5500/MANEJODOM/ejer2.html

### EJEMPLO HTML MANEJO DOM

Menú de Opciones  
**UNO** DOS TRES

#### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Ingrese Nombre  Ingrese Apellido

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI

→

← → ↻ 127.0.0.1:5500/MANEJODOM/ejer2.html

### EJEMPLO HTML MANEJO DOM

Menú de Opciones  
**UNO** DOS TRES

#### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Ingrese Nombre  Ingrese Apellido

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI

Así como podemos asignarle una clase también podemos obtener la clase que tenga o lista de clases.

## Ejemplo uso de style

```
const titulo = document.querySelector("h2")
titulo.style.backgroundColor= "indigo"
titulo.style.color="#00ff00"
titulo.style.padding="25px"
titulo.style.paddingLeft="25px"
titulo.style.paddingTop="25px"
titulo.style["paddingTop"]="25px"
titulo.style["padding-top"]="25px"
tittitulole.style.textAlign="center"
```

## Resultado

← → ↻ 127.0.0.1:5500/MANEJODOM/ejer2.html

### EJEMPLO HTML MANEJO DOM

Menú de Opciones  
**UNO** DOS TRES

#### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Ingrese Nombre  Ingrese Apellido

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI



## Manejo de eventos desde Javascript

### ¿Qué es un Evento?

Un evento no es más que una «notificación» de que una característica relevante acaba de suceder. Como desarrolladores, nuestro objetivo es preparar nuestro código para que **cuando ocurra un determinado evento**, se lleve a cabo una **funcionalidad asociada**. Este proceso se llama **escuchar eventos**. De esta forma, podemos preparar nuestra página o aplicación para que cuando ocurran ciertos eventos (que no podemos predecir de otra forma), reaccionen a ellos.

Existen varias formas de gestionar eventos sin necesidad de hacerlo desde nuestro .html mediante un atributo en línea. A continuación explicamos dos formas.

- **addEventListener():** La forma preferida de trabajar con eventos desde Javascript es utilizar el evento `.addEventListener()` sobre nuestro elemento particular
- **onNombreEvento:** El BOM (Browser Object Model) es un mecanismo tradicional de los navegadores para acceder a ciertos elementos a través de propiedades Javascript. La idea es la misma que vimos al utilizar listeners, sólo que en esta ocasión haremos uso de una propiedad Javascript, a la que le asignaremos la función con el código asociado

### Algunos Eventos:

- click
- mouseover
- mouseout
- change





## Ejemplo evento click desde Javascript

### EJEMPLO HTML MANEJO DOM

Menú de Opciones

**UNO** DOS TRES

#### LISTA EJEMPLO

- First
- Dos
- Tres
- Cuatro

Ingrese Nombre

Ingrese Apellido

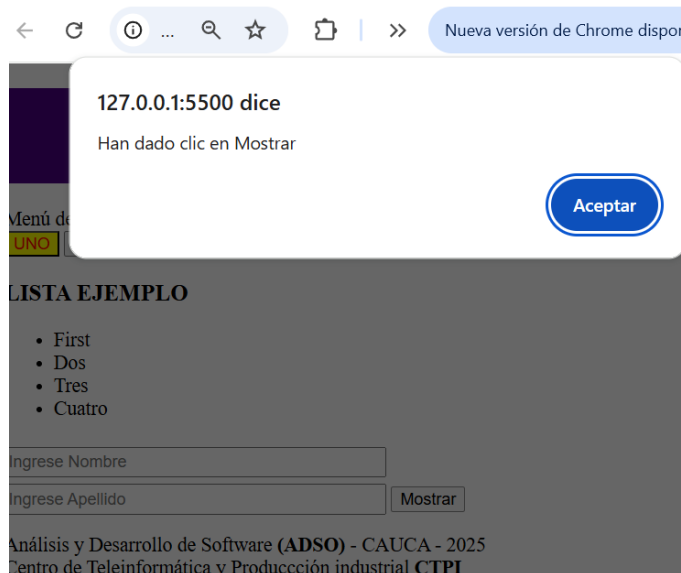
Mostrar

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI

Vamos a realizar una tarea al dar clic en el botón que dice **Mostrar**

```
const btnMostrar = document.querySelector('#btnMostrar')  
//primer forma uso programar evento click  
Tabnine | Edit | Test | Explain | Document  
btnMostrar.onclick={()=>{  
    alert("Han dado clic en Mostrar")  
}}
```

Prueba:





## Segunda forma

```
Tabnine | Edit | Test | Explain | Document  
btnMostrar.addEventListener("click",()=>{  
    alert("Han dado clic en Mostrar")  
})
```

Ahora vamos a programar el evento **change**. Cambiando el texto que se ingrese en las cajas de texto para que si se ingresa en minúscula lo pase a mayúscula.

```
elementosPorAtributo.forEach(elemento => {  
    elemento.addEventListener("change",()=>{  
        elemento.value = elemento.value.toUpperCase()  
    })  
})
```

## Resultado

← → ↺ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

---

**EJEMPLO HTML MANEJO D**

Menú de Opciones  
**UNO** DOS TRES

**LISTA EJEMPLO**

- First
- Dos
- Tres
- Cuatro

MONICA

GALINDO

Mostrar

Análisis y Desarrollo de Software (ADSO) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial CTPI



Por último vamos a programar el evento de pasar el mouse por encima del botón que dice **UNO** y cuando pase que cambie el estilo del botón y cuando ya no este por encima vuelva a colocar el estilo que tenía. Aquí se utilizan los eventos **mouseover** y **mouseout**.

```
Tabnine | Edit | Test | Explain | Document
botonUno.addEventListener("mouseover",()=>{
    botonUno.style.color="white"
    botonUno.style.backgroundColor="green"
})
```

```
Tabnine | Edit | Test | Explain | Document
botonUno.addEventListener("mouseout",()=>{
    botonUno.style.color="red"
    botonUno.style.backgroundColor="yellow"
})
```

## Resultado

← → ↻ ⓘ 127.0.0.1:5500/MANEJODOM/ejer2.html

**EJEMPLO HTML MANEJO DOM**

Menú de Opciones

**UNO** DOS TRES

**LISTA EJEMPLO**

- First
- Dos
- Tres
- Cuatro

Mostrar

Análisis y Desarrollo de Software (**ADSO**) - CAUCA - 2025  
Centro de Teleinformática y Producción industrial **CTPI**



**Nota:** Es importante la revisión de las referencias ya que en el material solo se han realizado

**Referencias:**

1. Tutorial de Javascript Moderno <https://es.javascript.info/>
2. Tutorial bootstrap w3schools: <https://www.w3schools.com/bootstrap5/>
3. Sitio oficial librería Bootstrap: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
4. Curso internet: <https://lenguajejs.com/dom/>
5. Curso internet: <https://lenguajejs.com/eventos/>
6. JavaScript para Manipulación del DOM - Curso con Proyectos  
<https://www.youtube.com/watch?v=koiPxFFiqJ4>

**CONTROL DEL DOCUMENTO**

	Nombre	Cargo	Dependencia	Fecha
<b>Autor (es)</b>	César Marino Cuéllar Chacón	Instructor	CTPI-CAUCA	10-06-2025