



**PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS**

**MÓDULOS EN JAVASCRIPT**

<b>Presentado a:</b>	Instructor César Marino Cuéllar Chacón
<b>Por Aprendiz:</b>	<b>Melva Cerón Buitrón</b>
<b>Ficha:</b>	3064975
<b>Competencia:</b>	Diseñar la solución de software de acuerdo con procedimientos y requisitos técnicos
<b>Resultado de Aprendizaje:</b>	Verificar los entregables de la fase de diseño del software de acuerdo con lo establecido en el informe de análisis

Tecnólogo en Análisis y Desarrollo de Software  
Servicio Nacional de Aprendizaje SENA  
Centro de Teleinformática y Producción Industrial  
Regional Cauca

Popayán, día 8 de 09 del año 2025



**PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS**

## **Tabla de Contenido**

<b>1. Ejercicio1</b>	<b>3</b>
1.1 Enunciado	3
1.2 Solución	4
<b>2. Ejercicio2</b>	<b>6</b>
2.1 Enunciado	6
2.2 Solución	6
<b>3. Ejercicio3</b>	<b>7</b>
3.1 Enunciado	7
3.2 Solución	7
<b>4. Ejercicio4</b>	<b>8</b>
4.1 Enunciado	9
<b>Escriba una función llamada nivelRiesgo que reciba dos parámetros: la edad de una persona y un valor booleano (true o false) que indique si la persona tiene enfermedades previas. La función debe retornar: • "Alto riesgo" si la persona tiene más de 60 años o tiene enfermedades previas. • "Riesgo moderado" si la persona tiene entre 40 y 60 años y tiene enfermedades previas. • "Bajo riesgo" en cualquier otro caso. Uso de módulos: • Exporta la función nivelRiesgo en un archivo utilidades.js. • En un archivo main.js, importa la función nivelRiesgo y pruébala con diferentes combinaciones de edad y enfermedades.</b>	
4.2 Solución	9
<b>5. Ejercicio5</b>	<b>10</b>
5.1 Enunciado	10
<b>Escriba una función llamada esBisiesto que reciba un año y determine si es bisiesto. Un año es bisiesto si: • Es divisible por 4, pero no es divisible por 100, a menos que también sea divisible por 400. La función debe retornar: • "Es bisiesto" si el año es bisiesto. • "No es bisiesto" si el año no lo es. Uso de módulos: • Exporta la función esBisiesto en un archivo utilidades.js. • En un archivo main.js, importa la función esBisiesto y pruébala con diferentes años.</b>	
5.2 Solución	10
<b>6. Ejercicio6</b>	<b>11</b>



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

6.1	Enunciado .....	11
7.	Ejercicio7.....	13
7.1	Enunciado .....	13
8.	Ejercicio8.....	18
9.	Ejercicio9.....	24
10	Bibliografía.....	32

### 1. Ejercicio1

#### 1.1 Enunciado

1. Crear una función llamada en Rango que reciba un número y determine si está en el rango entre 10 y 50 (inclusive). La función debe retornar: • "Está en el rango" si el número está entre 10 y 50 (inclusive). • "Fuera del rango" si el



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

número no está en ese rango. Uso de módulos: • Exporta la función en Rango en un archivo utilidades.js. • En un archivo main.js, importa la función en Rango y pruébala con diferentes números.

### 1.2 Solución

```
JS utilidades.js X JS main.js <> index.html 1
JS utilidades.js > ...
1
2 export function enRango(numero) {
3   if (numero >= 10 && numero <= 50) {
4     return "Está en el rango";
5   } else {
6     return "Fuera del rango";
7   }
8 }
```

```
JS utilidades.js JS main.js <> index.html 1
JS main.js
1 import { enRango } from "../utilidades.js";
2
3 // Pruebas enRango
4 console.log("---- EJERCICIO 1 ----");
5 console.log("Número 5:", enRango(5)); // Fuera del rango
6 console.log("Número 10:", (alias) enRango(numero: any): "Está en el rango" | "Fuera del rango"
7 console.log("Número 25:", import enRango
8 console.log("Número 50:", enRango(50)); // Está en el rango
9 console.log("Número 60:", enRango(60)); // Fuera del rango
10 |
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS utilidades.js JS main.js <> index.html 1 ●
<> index.html > ...
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4    <meta charset="UTF-8">
5    <title>Prueba Ejercicios</title>
6  </head>
7  <body>
8    <h2>Prueba de ejercicios en la consola</h2>
9    <script type="module" src="main.js"></script>
10 </body>
11 </html>
12
```

```
PROBLEMAS 1 SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\melva\OneDrive\Escritorio\javascript\modulos\taller\ejercicio1_rango> node main.js
---- EJERCICIO 1 ----
Número 5: Fuera del rango
Número 10: Está en el rango
Número 25: Está en el rango
Número 50: Está en el rango
Número 60: Fuera del rango
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### 2. Ejercicio2

#### 2.1 Enunciado

2. Escribir una función llamada `calcularDescuento` que reciba el precio de un producto y calcule

el descuento aplicable de acuerdo a la siguiente lógica:

- Si el precio es mayor a 1000, el descuento es del 20%.
- Si el precio es entre 500 y 1000, el descuento es del 10%.
- Si el precio es menor a 500, no se aplica descuento.

La función debe retornar el precio final después de aplicar el descuento.

Uso de módulos:

- Exporta la función `calcularDescuento` en un archivo `utilidades.js`.
- En un archivo `main.js`, importa la función `calcularDescuento` y pruébala con diferentes

precios.

#### 2.2 Solución

```
JS utilidades.js X JS main.js
JS utilidades.js > calcularDescuento
1 // Ejercicio 2: calcularDescuento
2 export function calcularDescuento(precio, porcentaje) {
3   let descuento = (precio * porcentaje) / 100;
4   let precioFinal = precio - descuento;
5   return { descuento, precioFinal };
6 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS utilidades.js JS main.js X
JS main.js
1  import { calcularDescuento } from "./utilidades.js";
2
3  // ---- EJERCICIO 2 ----
4  console.log("---- EJERCICIO 2 ----");
5  console.log("Precio: 1000, Descuento: 10%");
6  console.log(calcularDescuento(1000, 10));
7
8  console.log("Precio: 2500, Descuento: 25%");
9  console.log(calcularDescuento(2500, 25));
10
11 console.log("Precio: 500, Descuento: 5%");
12 console.log(calcularDescuento(500, 5));
```

### 3. Ejercicio3

#### 3.1 Enunciado

3. Escriba una función llamada `categorialMC` que reciba el índice de masa corporal (IMC) de una persona y determine su categoría: • "Bajo peso" si el IMC es menor a 18.5. • "Normal" si el IMC está entre 18.5 y 24.9. • "Sobrepeso" si el IMC está entre 25 y 29.9. • "Obesidad" si el IMC es 30 o mayor. Uso de módulos: • Exporta la función `categorialMC` en un archivo `utilidades.js`. • En un archivo `main.js`, importa la función `categorialMC` y pruébala con diferentes valores de IMC.

#### 3.2 Solución



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS utilidades.js X JS main.js
JS utilidades.js > categoriaIMC
1
2 export function categoriaIMC(peso, altura) {
3     let imc = peso / (altura * altura);
4     if (imc < 18.5) return "Bajo peso";
5     if (imc < 25) return "Normal";
6     if (imc < 30) return "Sobrepeso";
7     return "Obesidad";
8 }
```

```
JS utilidades.js X JS main.js X
JS main.js
1 import { categoriaIMC } from "../utilidades.js";
2
3 // ---- EJERCICIO 3 ----
4 console.log("---- EJERCICIO 3 ----");
5 console.log("Peso: 50kg, Altura: 1.70m →", categoriaIMC(50, 1.70)); // Bajo peso
6 console.log("Peso: 70kg, Altura: 1.70m →", categoriaIMC(70, 1.70)); // Normal
7 console.log("Peso: 85kg, Altura: 1.70m →", categoriaIMC(85, 1.70)); // Sobrepeso
8 console.log("Peso: 100kg, Altura: 1.70m →", categoriaIMC(100, 1.70)); // Obesidad
```

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\melva\OneDrive\Escritorio\javascript\modulos\taller\ejercicio3> node main.js
---- EJERCICIO 3 ----
Peso: 50kg, Altura: 1.70m → Bajo peso
Peso: 70kg, Altura: 1.70m → Normal
Peso: 85kg, Altura: 1.70m → Sobrepeso
Peso: 100kg, Altura: 1.70m → Obesidad
```

### 4. Ejercicio4





## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### 4.1 Enunciado

Escriba una función llamada `nivelRiesgo` que reciba dos parámetros: la edad de una persona y un valor booleano (`true` o `false`) que indique si la persona tiene enfermedades previas. La función debe retornar: • "Alto riesgo" si la persona tiene más de 60 años o tiene enfermedades previas. • "Riesgo moderado" si la persona tiene entre 40 y 60 años y tiene enfermedades previas. • "Bajo riesgo" en cualquier otro caso. Uso de módulos: • Exporta la función `nivelRiesgo` en un archivo `utilidades.js`. • En un archivo `main.js`, importa la función `nivelRiesgo` y pruébala con diferentes combinaciones de edad y enfermedades.

### 4.2 Solución

```
JS utilidades.js X JS main.js 1
JS utilidades.js > nivelRiesgo
1 // Ejercicio 4: nivelRiesgo
2 export function nivelRiesgo(edad) {
3     if (edad < 18) return "Bajo";
4     if (edad <= 40) return "Medio";
5     return "Alto";
6 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS utilidades.js  JS main.js 1 X
JS main.js
1  import { nivelRiesgo } from "./utilidades.js";
2
3
4  console.log("---- EJERCICIO 4 ----");
5  console.log("Edad 15 →", nivelRiesgo(15)); // Bajo
6  console.log("Edad 25 →", nivelRiesgo(25)); // Medio
7  console.log("Edad 40 →", nivelRiesgo(40)); // Medio
8  console.log("Edad 50 →", nivelRiesgo(50)); // Alto
```

```
PROBLEMAS 1  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS C:\Users\melva\OneDrive\Escritorio\javascript\modulos\taller\ejercicio4> node main.js
---- EJERCICIO 4 ----
Edad 15 → Bajo
Edad 25 → Medio
Edad 40 → Medio
Edad 50 → Alto
```

## 5. Ejercicio5

### 5.1 Enunciado

Escriba una función llamada `esBisiesto` que reciba un año y determine si es bisiesto. Un año es bisiesto si: • Es divisible por 4, pero no es divisible por 100, a menos que también sea divisible por 400. La función debe retornar: • "Es bisiesto" si el año es bisiesto. • "No es bisiesto" si el año no lo es. Uso de módulos: • Exporta la función `esBisiesto` en un archivo `utilidades.js`. • En un archivo `main.js`, importa la función `esBisiesto` y pruébala con diferentes años.

### 5.2 Solución



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS utilidades.js • JS main.js
JS utilidades.js > esBisiesto
1 // esBisiesto
2 export function esBisiesto(anio) {
3   return (anio % 4 === 0 && anio % 100 !== 0) || (anio % 400 === 0);
4 }
```

```
JS utilidades.js • JS main.js ×
JS main.js
1 import { esBisiesto } from "../utilidades.js";
2 console.log("---- EJERCICIO 5 ----");
3 console.log("Año 2024 →", esBisiesto(2024)); // true (bisiesto)
4 console.log("Año 2023 →", esBisiesto(2023)); // false
5 console.log("Año 2000 →", esBisiesto(2000)); // true (bisiesto)
6 console.log("Año 1900 →", esBisiesto(1900)); // false
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
PS C:\Users\melva\OneDrive\Escritorio\javascript\modulos\taller\ejercicio5> node main.js
---- EJERCICIO 5 ----
Año 2024 → true
Año 2023 → false
Año 2000 → true
Año 1900 → false
```

## 6. Ejercicio6

### 6.1 Enunciado

Escribe una función llamada **esElegibleParaPrestamo** que reciba dos parámetros: el salario mensual de una persona y su puntaje de crédito. La función debe retornar: • "Elegible para préstamo" si el salario es mayor a 3000000 y el puntaje de crédito es mayor a 650. • "No elegible para préstamo"



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

en cualquier otro caso. **Uso de módulos:**

- Exporta la función `esElegibleParaPrestamo` en un archivo `utilidades.js`.
- En un archivo `main.js`, importa la función `esElegibleParaPrestamo` y pruébala con diferentes valores de salario y puntaje de crédito.

### 6.2 Solución

```
JS utilidades.js X JS main.js
JS utilidades.js > ...
1 //esElegibleParaPrestamo
2 export function esElegibleParaPrestamo(edad, ingresos, deudas) {
3     if (edad >= 18 && ingresos >= 1000 && deudas === false) {
4         return true;
5     }
6     return false;
7 }
```

```
JS utilidades.js X JS main.js X
JS main.js
1 import { esElegibleParaPrestamo } from "../utilidades.js";
2
3 console.log("---- EJERCICIO 6 ----");
4 console.log("Edad 20, Ingresos 1500, Sin deudas →", esElegibleParaPrestamo(20, 1500, false));
5 console.log("Edad 17, Ingresos 2000, Sin deudas →", esElegibleParaPrestamo(17, 2000, false));
6 console.log("Edad 25, Ingresos 800, Sin deudas →", esElegibleParaPrestamo(25, 800, false));
7 console.log("Edad 30, Ingresos 1500, Con deudas →", esElegibleParaPrestamo(30, 1500, true));
```

```
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS
PS C:\Users\melva\OneDrive\Escritorio\javascript\modulos\taller\ejercicio6> node main.js
---- EJERCICIO 6 ----
Edad 20, Ingresos 1500, Sin deudas → true
Edad 17, Ingresos 2000, Sin deudas → false
Edad 25, Ingresos 800, Sin deudas → false
Edad 30, Ingresos 1500, Con deudas → false
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### 7. Ejercicio7

#### 7.1 Enunciado

Crear un archivo en formato json llamado `libros.json` que contenga objetos de tipo libro que tengan los siguientes atributos:

- a. Título
- b. Autor
- c. Número páginas
- d. Editorial

e. Idioma Hacer una función llamada `consultarLibrosPorPalabraClaveTitulo` que reciba como parámetro una palabra. La función debe retornar un arreglo con todos los libros que en su título contenga la palabra que recibe como parámetro. Hacer una función llamada `consultarLibrosPaginas` que crea un arreglo de objetos que contienen dos atributos así:

título del libro y número de páginas de todos los libros existentes. Uso de módulos:

- En un archivo `utilidades.js` importa el archivo `libros.json`
- Exporta la función `consultarLibrosPorPalabraClaveTitulo` en un archivo `utilidades.js`.
- Exporta la función `consultarLibrosPaginas` en un archivo `utilidades.js`.
- En un archivo `main.js`, importa las funciones

`consultarLibrosPorPalabraClaveTitulo`, `consultarLibrosPaginas` pruébala con diferentes valores de salario y puntaje de crédito.

#### 6.2 Solución



PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS

```
{ } libros.json X JS utilidades.js JS main.js <> index.html 1

{ } libros.json > { } 1 > titulo

1  [
2  {
3    "titulo": "Cien años de soledad",
4    "autor": "Gabriel García Márquez",
5    "paginas": 417,
6    "editorial": "Sudamericana",
7    "idioma": "Español"
8  },
9  {
10   ✨ "titulo": "Don Quijote de la Mancha",
11     "autor": "Miguel de Cervantes",
12     "paginas": 863,
13     "editorial": "Francisco de Robles",
14     "idioma": "Español"
15   },
16   {
17     "titulo": "The Great Gatsby",
18     "autor": "F. Scott Fitzgerald",
19     "paginas": 218,
20     "editorial": "Charles Scribner's Sons",
21     "idioma": "Inglés"
22   },
23   {
24     "titulo": "La ciudad y los perros",
25     "autor": "Mario Vargas Llosa",
26     "paginas": 376,
27     "editorial": "Seix Barral",
28     "idioma": "Español"
29   }
30 ]
```



PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS

```
{ } libros.json  JS utilidades.js X  JS main.js  <> index.html 1

JS utilidades.js > consultarLibrosPaginas
1  let libros = [];
2
3  // Cargar libros desde el JSON usando fetch
4  export async function cargarLibros() {
5      const respuesta = await fetch("./libros.json");
6      libros = await respuesta.json();
7      return libros;
8  }
9
10 // Buscar libros por palabra clave en el título
11 export function consultarLibrosPorPalabraClaveTitulo(palabra) {
12     return libros.filter(libro =>
13         libro.titulo.toLowerCase().includes(palabra.toLowerCase())
14     );
15 }
16
17 // Consultar títulos y páginas de todos los libros
18 export function consultarLibrosPaginas() {
19     return libros.map(libro => ({
20         titulo: libro.titulo,
21         paginas: libro.paginas
22     }));
23 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
{ } libros.json  JS utilidades.js  JS main.js  X  <> index.html 1

JS main.js > ...
1  import {
2    cargarLibros,
3    consultarLibrosPorPalabraClaveTitulo,
4    consultarLibrosPaginas
5  } from "../utilidades.js";
6
7  // Esperamos a que carguen los libros antes de activar los botones
8  cargarLibros().then(() => {
9    console.log("✅ Libros cargados correctamente");
10
11    // ---- BOTÓN BUSCAR ----
12    document.querySelector("#btnBuscar").addEventListener("click", () => {
13      let palabra = document.querySelector("#palabra").value.trim();
14      let resultados = consultarLibrosPorPalabraClaveTitulo(palabra);
15
16      let ul = document.querySelector("#resultadoBusqueda");
17      ul.innerHTML = "";
18
19      if (resultados.length === 0) {
20        ul.innerHTML = `<li class="list-group-item text-danger">No se encontraron libros</li>`;
21      } else {
22        resultados.forEach(libro => {
23          ul.innerHTML += `<li class="list-group-item">
24            <b>${libro.titulo}</b> - ${libro.autor} (${libro.idioma})
25          </li>`;
26        });
27      }
28    });
29
30    // ---- BOTÓN MOSTRAR TODOS ----
31    document.querySelector("#btnListar").addEventListener("click", () => {
32      let libros = consultarLibrosPaginas();
33      let tbody = document.querySelector("#tablaLibros");
34      tbody.innerHTML = "";
35
36      libros.forEach(libro => {
37        tbody.innerHTML += `
38          <tr>
39            <td>${libro.titulo}</td>
40            <td>${libro.paginas}</td>
41          </tr>
42          `;
43      });
44    });
45  });
```





## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
libros.json JS utilidades.js JS main.js index.html 1 X
index.html > html
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Gestión de Libros</title>
6   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
7 </head>
8 <body class="container py-4">
9
10  <h2 class="text-center fw-bold"> Gestión de Libros</h2>
11
12  <!-- Buscar por palabra -->
13  <div class="mt-4">
14    <label for="palabra" class="form-label">Buscar libros por palabra en el título:</label>
15    <input type="text" id="palabra" class="form-control" placeholder="Ej: ciudad, the...">
16    <button class="btn btn-primary mt-2" id="btnBuscar">Buscar</button>
17  </div>
18
19  <!-- Resultados búsqueda -->
20  <div class="mt-4">
21    <h4>Resultados de búsqueda:</h4>
22    <ul id="resultadoBusqueda" class="list-group"></ul>
23  </div>
24
25  <!-- Consultar títulos y páginas -->
26  <div class="mt-5">
27    <h4>Títulos y número de páginas:</h4>
28    <table class="table table-bordered">
29      <thead class="table-dark">
30        <tr>
31          <th>Título</th>
32          <th>Páginas</th>
33        </tr>
34      </thead>
35      <tbody id="tablaLibros"></tbody>
36    </table>
37    <button class="btn btn-success" id="btnListar">Mostrar todos</button>
38  </div>
39
40  <script type="module" src="./main.js"></script>
41 </body>
42 </html>
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### Gestión de Libros

Buscar libros por palabra en el título:

Buscar

Resultados de búsqueda:

Títulos y número de páginas:

Título

Páginas

Mostrar todos

### Gestión de Libros

Buscar libros por palabra en el título:

Buscar

Resultados de búsqueda:

Cien años de soledad - Gabriel García Márquez (Español)

Títulos y número de páginas:

Título	Páginas
Cien años de soledad	417
Don Quijote de la Mancha	863
The Great Gatsby	218
La ciudad y los perros	376

Mostrar todos

## 8. Ejercicio8

### 8.1 Enunciado

Proyecto Alcancia. En la alcancia es posible guardar monedas de las siguientes denominaciones: \$200, \$500 y de \$1000. No se guardan ni billetes ni monedas de



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

otras denominaciones. Al dueño de la alcancía le parece muy útil conocer cuánto tiene en la alcancía sin necesidad de romperla, es más, él quiere conocer cuántas monedas tiene en cada denominación para así romper la alcancía sólo cuando quiera disponer de todo su dinero ahorrado. Por lo anterior se necesita crear una aplicación web que le permita simular el comportamiento de la alcancía.

La aplicación debe permitir:

- Agregar una moneda de una de las denominaciones indicadas.
- Contar cuántas monedas tiene de cada denominación.
- Calcular el total de dinero ahorrado.
- Romper la alcancía vaciando su contenido.
- Comenzar una nueva alcancía.

La aplicación debe hacerse orientada a objetos. Para ello usted debe crear una clase llamada Alcancia, la cual tiene 3 atributos que representan la cantidad de monedas de cada una de las denominaciones. Adicionalmente la alcancía tiene que tener unos métodos o funciones que permiten realizar las tareas mencionadas anteriormente.

Uso de módulos

- Hacer un export default a la clase Alcancia en el archivo alcancia.js
- En un archivo main.js importar la clase Alcancia
- En el archivo main.js realizar todas las operaciones.
- Crear un archivo html para interactuar con el usuario el cual debe crear una etiqueta script para que incorpore el archivo main.js como type=module

### 8.2 Solución



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS alcancia.js X <> <!DOCTYPE html> Untitled-1 JS mainM.js
C: > Users > melva > OneDrive > Escritorio > javascript > modulos > taller > Alcancia > JS alcancia.js > Alcancia
1  export default class Alcancia {
2      /**
3       * contrutor de la Alcancia
4       * inicia vacia
5       */
6      constructor() {
7          this.moneda200 = 0;
8          this.moneda500 = 0;
9          this.moneda1000 = 0;
10     }
11
12     agregarMoneda200() {
13         this.moneda200++;
14     }
15
16     agregarMoneda500() {
17         this.moneda500++;
18     }
19
20     agregarMoneda1000() {
21         this.moneda1000++;
22     }
23
24     /**
25     * calcular la cantidad de dinero
26     * ahorrado en la alcancia
27     * @returns int
28     */
29     calcularTotalMonedas() {
30         const total = (this.moneda200 * 200) +
31             (this.moneda500 * 500) +
32             (this.moneda1000 * 1000);
33         return total;
34     }
35 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS alcancia.js  index.html 5 x JS mainM.js
C: > Users > melva > OneDrive > Escritorio > javascript > modulos > taller > Alcancia > index.html > html
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   <title>Document</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
8   <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
9 </head>
10 <body>
11   <div class="container">
12     <div class="row">
13       <div class="col">
14         
15       </div>
16       <div class="col">
17         
18       </div>
19       <div class="col">
20         
21       </div>
22     </div>
23
24     <!-- Aquí estaba el error -->
25     <div class="col-6" style="margin-top:20px">
26       <div class="mt-2">
27         <label>Monedas 200:</label>
28         <input type="text" id="txtMonedas200" class="form-control" disabled/>
29       </div>
30       <div class="mt-2">
31         <label>Monedas 500:</label>
32         <input type="text" id="txtMonedas500" class="form-control" disabled/>
33       </div>
34       <div class="mt-2">
35         <label>Monedas 1000:</label>
36         <input type="text" id="txtMonedas1000" class="form-control" disabled/>
37       </div>
38
39       <div class="mt-2">
40         <label>Total Alcancia:</label>
41         <input type="text" id="txtTotalAlcancia" class="form-control" disabled/>
42       </div>
43       <div class="mt-2">
44         <button class="btn btn-secondary" id="btnRomper">Romper...</button>
45       </div>
46     </div>
47
48     <script type="module" src="./mainM.js"></script>
49 </body>
50 </html>
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS alcancia.js  <> index.html 5  JS mainM.js  X
C: > Users > melva > OneDrive > Escritorio > javascript > modulos > taller > Alcancia > JS mainM.js > ...
1  import Alcancia from "./alcancia.js"
2
3  let miAlcancia = null
4
5  const crearAlcancia = () =>{
6      miAlcancia = new Alcancia()
7  }
8
9  const actualizarDatosInterfaz = () =>{
10     document.querySelector("#txtMonedas200").value=miAlcancia.moneda200
11     document.querySelector("#txtMonedas500").value=miAlcancia.moneda500
12     document.querySelector("#txtMonedas1000").value=miAlcancia.moneda1000
13     document.querySelector("#txtTotalAlcancia").value=miAlcancia.calcularTotalMonedas()
14 }
15
16 document.querySelector("#img200").addEventListener("click",()=>{
17     miAlcancia.agregarMoneda200()
18     actualizarDatosInterfaz()
19 })
20 document.querySelector("#img500").addEventListener("click",()=>{
21     miAlcancia.agregarMoneda500()
22     actualizarDatosInterfaz()
23 })
24
25 document.querySelector("#img1000").addEventListener("click",()=>{
26     miAlcancia.agregarMoneda1000()
27     actualizarDatosInterfaz()
28 })
29
30 const inicializarAlcancia=()=>{
31     crearAlcancia()
32     actualizarDatosInterfaz()
33 }
34
35
36 inicializarAlcancia()
37
38 // botón Romper
39 document.querySelector("#btnRomper").addEventListener("click", () => {
40     alert("Rompeste la alcancia\nAhorro total: $" + miAlcancia.calcularTotalMonedas())
41
42     // Reiniciar alcancia
43     crearAlcancia()
44     actualizarDatosInterfaz()
45 })
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS



Monedas 200:

Monedas 500:

Monedas 1000:

Total Alcancia:

Romper...



Monedas 200:

Monedas 500:


Monedas 1000:

Total Alcancia:

Romper...



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS



melvacb23.github.io dice

Rompiste la alcancía

Ahorro total: \$5400

Aceptar

Monedas 200:

Monedas 500:

Monedas 1000:

Total Alcancía:

Romper...



Monedas 200:

Monedas 500:

Monedas 1000:

Total Alcancía:

Romper...

### 9. Ejercicio9





PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS

### 9.1 Enunciado

Hacer una aplicación que permita gestionar sus contactos. Para ello debe crear una clase Contacto en un archivo llamado contacto.js. La clase debe tener los siguientes atributos:

- a. Identificación
- b. Nombre
- c. Apellido
- d. Correo
- e. Celular

Uso de módulos

- Hacer un export default a la clase Contacto en el archivo contacto.js
- En un archivo main.js importar la clase Contacto
- Crear un archivo llamado app.html con el código html de la interfaz anexa. Aquí se debe crear una etiqueta script de type=module llamando al archivo main.js
- En el archivo main.js se deben crear las funciones que le permita hacer las siguientes

operaciones:

- o Agregar un contacto a la agenda. Se debe mostrar en la tabla inferior. No puede haber más de un contacto con la misma identificación
- o Consultar contacto por identificación.
- o Actualizar Contacto
- o Eliminar Contacto



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### Agenda de Contactos

Identificación:

Nombre:

Apellido:

Correo:

Celular:

Agregar

Consultar

Actualizar

Eliminar

Id	Nombre	Apellido	Correo	Celular
11	Monik	Galindo	305555555	mgalindo@yahoo.com

## 9.2 Solución

```
JS contacto.js X JS main.js <> index.html
JS contacto.js > ...
1 export default class Contacto {
2   constructor(identificacion, nombre, apellido, correo, celular) {
3     this.identificacion = identificacion;
4     this.nombre = nombre;
5     this.apellido = apellido;
6     this.correo = correo;
7     this.celular = celular;
8   }
9 }
10
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
JS contacto.js JS main.js X <> index.html
JS main.js > limpiarFormulario
1  import Contacto from "../contacto.js";
2
3  let agenda = [];
4
5  // Referencias a los inputs
6  const idInput = document.querySelector("#identificacion");
7  const nombreInput = document.querySelector("#nombre");
8  const apellidoInput = document.querySelector("#apellido");
9  const correoInput = document.querySelector("#correo");
10 const celularInput = document.querySelector("#celular");
11
12 // Referencia a la tabla
13 const tabla = document.querySelector("#tablaContactos");
14
15 // Función para renderizar la tabla
16 function mostrarContactos() {
17     tabla.innerHTML = "";
18     agenda.forEach(contacto => {
19         tabla.innerHTML += `
20             <tr>
21                 <td>${contacto.identificacion}</td>
22                 <td>${contacto.nombre}</td>
23                 <td>${contacto.apellido}</td>
24                 <td>${contacto.celular}</td>
25                 <td>${contacto.correo}</td>
26             </tr>
27         `;
28     });
29 }
30
31 // Agregar contacto
32 document.querySelector("#btnAgregar").addEventListener("click", () => {
33     let id = idInput.value.trim();
34
35     if (agenda.some(c => c.identificacion === id)) {
36         alert("⚠ Ya existe un contacto con esta identificación");
37         return;
38     }
39     // Lógica para agregar el contacto a la agenda
40     agenda.push({
41         identificacion: id,
42         nombre: nombreInput.value,
43         apellido: apellidoInput.value,
44         celular: celularInput.value,
45         correo: correoInput.value
46     });
47     mostrarContactos();
48     // Limpiar los inputs
49     idInput.value = "";
50     nombreInput.value = "";
51     apellidoInput.value = "";
52     celularInput.value = "";
53     correoInput.value = "";
54 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
38     }
39
40     let nuevo = new Contacto(
41         id,
42         nombreInput.value.trim(),
43         apellidoInput.value.trim(),
44         correoInput.value.trim(),
45         celularInput.value.trim()
46     );
47
48     agenda.push(nuevo);
49     mostrarContactos();
50     limpiarFormulario();
51 });
52
53 // Consultar contacto
54 document.querySelector("#btnConsultar").addEventListener("click", () => {
55     let id = idInput.value.trim();
56     let contacto = agenda.find(c => c.identificacion === id);
57
58     if (contacto) {
59         nombreInput.value = contacto.nombre;
60         apellidoInput.value = contacto.apellido;
61         correoInput.value = contacto.correo;
62         celularInput.value = contacto.celular;
63     } else {
64         alert("✗ No se encontró el contacto");
65     }
66 });
67
68 // Actualizar contacto
69 document.querySelector("#btnActualizar").addEventListener("click", () => {
70     let id = idInput.value.trim();
71     let contacto = agenda.find(c => c.identificacion === id);
72
```



PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS

```
73     if (contacto) {
74         contacto.nombre = nombreInput.value.trim();
75         contacto.apellido = apellidoInput.value.trim();
76         contacto.correo = correoInput.value.trim();
77         contacto.celular = celularInput.value.trim();
78         mostrarContactos();
79         limpiarFormulario();
80     } else {
81         alert("✗ No se encontró el contacto para actualizar");
82     }
83 });
84
85 // Eliminar contacto
86 document.querySelector("#btnEliminar").addEventListener("click", () => {
87     let id = idInput.value.trim();
88     agenda = agenda.filter(c => c.identificacion !== id);
89     mostrarContactos();
90     limpiarFormulario();
91 });
92
93 // Limpiar formulario
94 function limpiarFormulario() {
95     idInput.value = "";
96     nombreInput.value = "";
97     apellidoInput.value = "";
98     correoInput.value = "";
99     celularInput.value = "";
100 }
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

```
15 main.js 16 main.js 17 index.html X
18 <!-->
19 <!-->
20 <!-->
21 <!-->
22 <!-->
23 <!-->
24 <!-->
25 <!-->
26 <!-->
27 <!-->
28 <!-->
29 <!-->
30 <!-->
31 <!-->
32 <!-->
33 <!-->
34 <!-->
35 <!-->
36 <!-->
37 <!-->
38 <!-- Botones -->
39 <div class="mb-3">
40 <button id="btnAgregar" class="btn btn-secondary">Agregar</button>
41 <button id="btnconsultar" class="btn btn-dark">Consultar</button>
42 <button id="btnActualizar" class="btn btn-info">Actualizar</button>
43 <button id="btnEliminar" class="btn btn-danger">Eliminar</button>
44 </div>
45
46 <!-- Tabla -->
47 <table class="table table-striped">
48 <thead class="table-dark">
49 <tr>
50 <th>Id</th>
51 <th>Nombre</th>
52 <th>Apellido</th>
53 <th>Celular</th>
54 <th>Correo</th>
55 </tr>
56 </thead>
57 <tbody id="tablaContactos"></tbody>
58 </table>
59
60 <script type="module" src="./main.js"></script>
61 </body>
62 </html>
```



## PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL FORMATO ENTREGA DE EVIDENCIAS

### Agenda de Contactos

Identificación:

Nombre:

Apellido:

Correo:

Celular:

Agregar

Consultar

Actualizar

Eliminar

Id	Nombre	Apellido	Celular	Correo
1029605328	juan david	campo	35346547567	fafsgfergdfgd



**PROCESO DE GESTIÓN DE LA FORMACIÓN PROFESIONAL INTEGRAL  
FORMATO ENTREGA DE EVIDENCIAS**

## **10 Bibliografía**

1. Material de apoyo, suministrado por el instructor Cesar Cuellar.
2. Tutorial de Javascript Moderno <https://es.javascript.info/> .
3. Tutorial de w3schools: <https://www.w3schools.com/js/>.
4. Curso en youtube:  
<https://www.youtube.com/watch?v=Z34BF9PCfYg&t=106s>.
5. Curso internet: <https://lenguajejs.com/javascript/>