Desafío Práctico 1
HTML

Kevin Alfredo Romero Deras **RD220238**

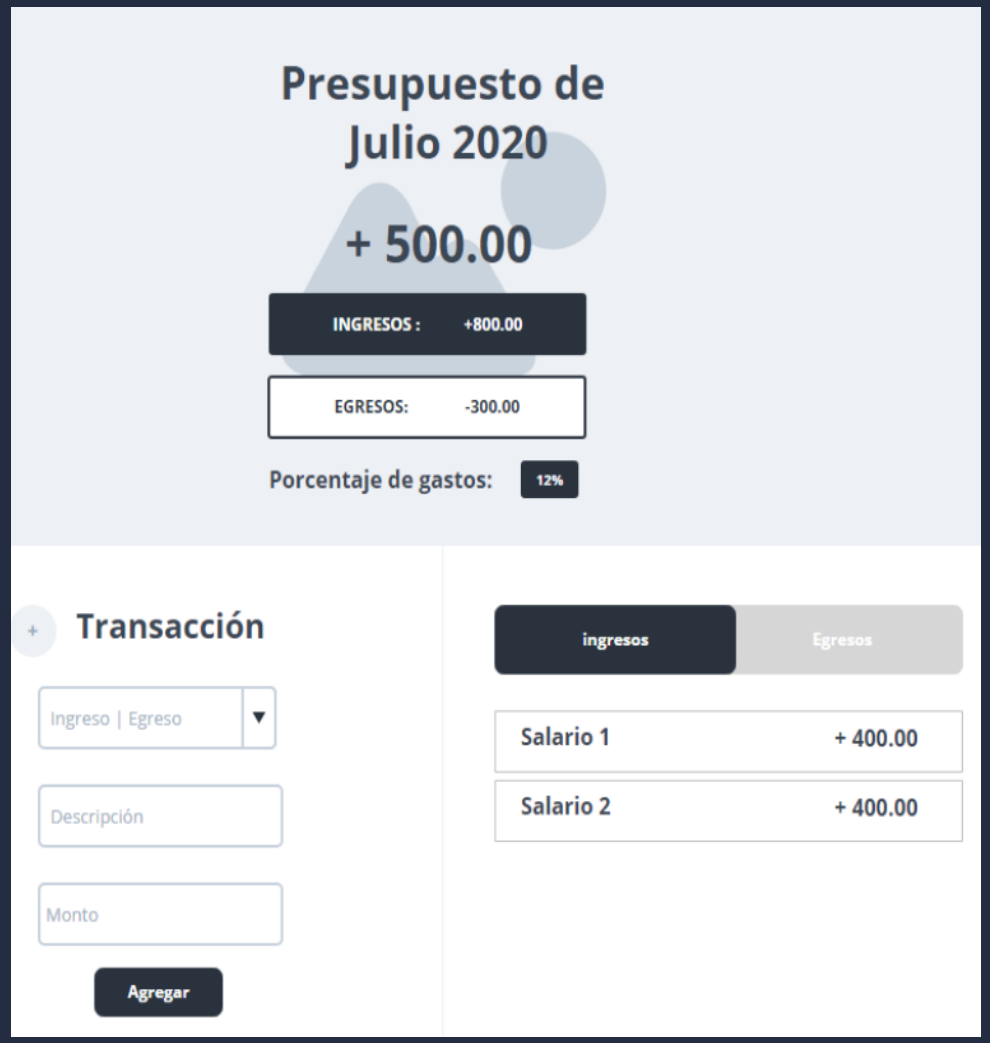
Jorge Marvin Peña Roque **PR243380**

Desafío 1 : Aplicación de Presupuesto

Problema:

- Se desea realizar una aplicación en JavaScript que permita manejar el flujo de ingresos y egresos de una persona. De forma que, el usuario pueda valorar el porcentaje de gastos que ha tenido durante un mes basado en su ingreso, usando el siguiente diseño propuesto.

- Parte Gráfica



- Funcionalidades

Usando el Mockup anterior, se debe crear una aplicación JavaScript que maneje los objetos necesarios para el correcto funcionamiento de la aplicación. Considerando los siguientes puntos:

1. El mes en el título de la aplicación debe ser calculado basado en la fecha actual, por ejemplo, si la fecha actual es 23 de octubre de 2020, el título deberá ser: Presupuesto de octubre 2020.
2. El monto total disponible en el mes deberá ser calculado con base en las transacciones generadas. La suma de los ingresos menos la suma de los egresos.
3. Los cuadros: Ingresos y egresos, deberán mostrar las sumas de los tipos de transacciones respectivamente. En el caso del diseño, existen 2 transacciones de tipo Ingreso, lo cual suma 800.00, el monto que aparece en el cuadro de ingreso. Realizar el mismo proceso para los egresos.
    a. Para el caso de los egresos se debe calcular el porcentaje total de gastos (porcentaje de gastos) que se ha generado a partir de los egresos, tomando como ejemplo el diseño:

$$\%Egreso = \frac{TotalEgresos * 100}{TotalIngresos}$$

$$\%Egreso = \frac{300 * 100}{800}$$

$$\%Egreso = 11.60 \sim 12\%$$

    b. Se debe utilizar la fusión ***toFixed(2)*** para realizar el redondeo de las cifras.
4. En la sección de Transacción, considerar los siguientes puntos:
    a. El menú desplegable (Dropdonw) deberá incluir las opciones: Ingreso y Egreso.
    b. Agregar campo de descripción
    c. Campo para monto, que deberá ser validado para solo aceptar números.
    d. Al agregar una nueva transacción, deberá aparecer en la lista de ingresos o egresos respectivamente.
5. En la sección de detalles, considerar los siguientes puntos:
    a. Existirá 2 pestañas (tabs) con las opciones Ingresos y egresos, que, al hacer clic sobre alguna de ellas, deberá mostrar la lista de las transacciones realizadas.
    b. Para el caso del tab de ingresos, deberá mostrarse el texto de descripción, junto con el monto del ingreso.
    c. Para el caso del tab de egresos, deberá mostrarse el texto de descripción , junto con el monto del egreso y el calculo del porcentaje de gasto que representa la transacción

Solución de Requerimiento

• Parte Gráfica

Para la generación de la **parte Gráfica (Front-End)** se utilizará Lenguaje de Marcado de Hipertexto (HTML) y apoyándonos para el estilo con CSS (Hojas es estilo en Cascada). En la cual el cliente podrá visualizar la información, interactuar y verificar los resultados de los datos ingresados de forma amigable, en este caso "Presupuesto del mes presente, ingresos, egresos, totales y porcentajes respectivos".

• Parte de Funcionalidades

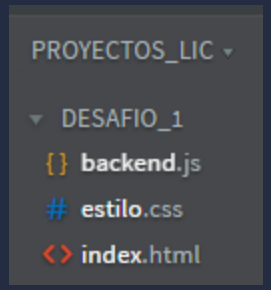
Para el desarrollo o implementación codificada de la parte de **Funcionalidades (Back-End)** se utilizará Lenguaje de programación JS (JavaScript) puro. Dicho desarrollo consta de funciones para todos los puntos específicos descritas en la documentación del desafío. Las herramientas utilizadas para el desarrollo:

- IDE Brackets - HTML
- Lenguaje JS - CSS

Arquitectura de Proyecto

Se divide en tres archivos:

- index.html
- estilo.css
- backend.js



Generación de Interfaz de usuario

- **index.html**, genera cada uno de los componentes como : botones, listas seleccionables, y contenedores de información. Además, tienen ciertos códigos para invocar propiedades para los componentes a través de archivo estilo.css, así como Id que permiten llamar variables cargadas por las funciones de archivo backend.js
- Backend.js es cargado por :

```
<script src="backend.js"></script>
```

- Estilo.css es cargado por:

```
<link rel="stylesheet" type="text/css" href="estilo.css">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="estilo.css">
    <title id="app-title">Presupuesto</title>
</head>

<body>
    <div class="container">
        <div class="message" > Presupuesto de
        <p id="elemento-fecha"></p> <p class="message2" id="elemento-saldoTotal"></p>
         </div>
        <div class="c1">INGRESOS: <p id="elemento-ingresoTotal"></p> </div>
        <div class="c2">EGRESOS: <p id="elemento-egresoTotal"></p> </div>
        <div class="c3">
            <div class="message3"> Porcentaje de gastos: </div>
            <div class="porcentage" id="porcentaje-gastos"></div>
        </div>
    </div>
    <div class="container2">
        <form class="inside1" id="transaction-form">
            <div class="m1">Transacción</div>
            <!-- El menú desplegable (Dropdonw) -->
            <select class="c4" id="transaction-type">
                <option value="ingreso">Ingreso</option>
                <option value="egreso">Egreso</option>
            </select>
            <input class="c5" type="text" id="description" placeholder="Descripción" />
            <input class="c6" type="number" id="amount" placeholder="Monto" step="0.01" />
            <button class="c7" type="submit">Agregar</button>
        </form>
        <div class="inside2">
            <div class="buttoncontainer">
                <button class="c8" id="show-incomes">Ingresos</button>
                <button class="c9" id="show-expenses">Egresos</button>
            </div>
            <div class="data" id="data-list">  </div>
        </div>
    </div>
    <script src="backend.js"></script>
```

Estilo de Interfaz Gráfica

- **estilo.css**

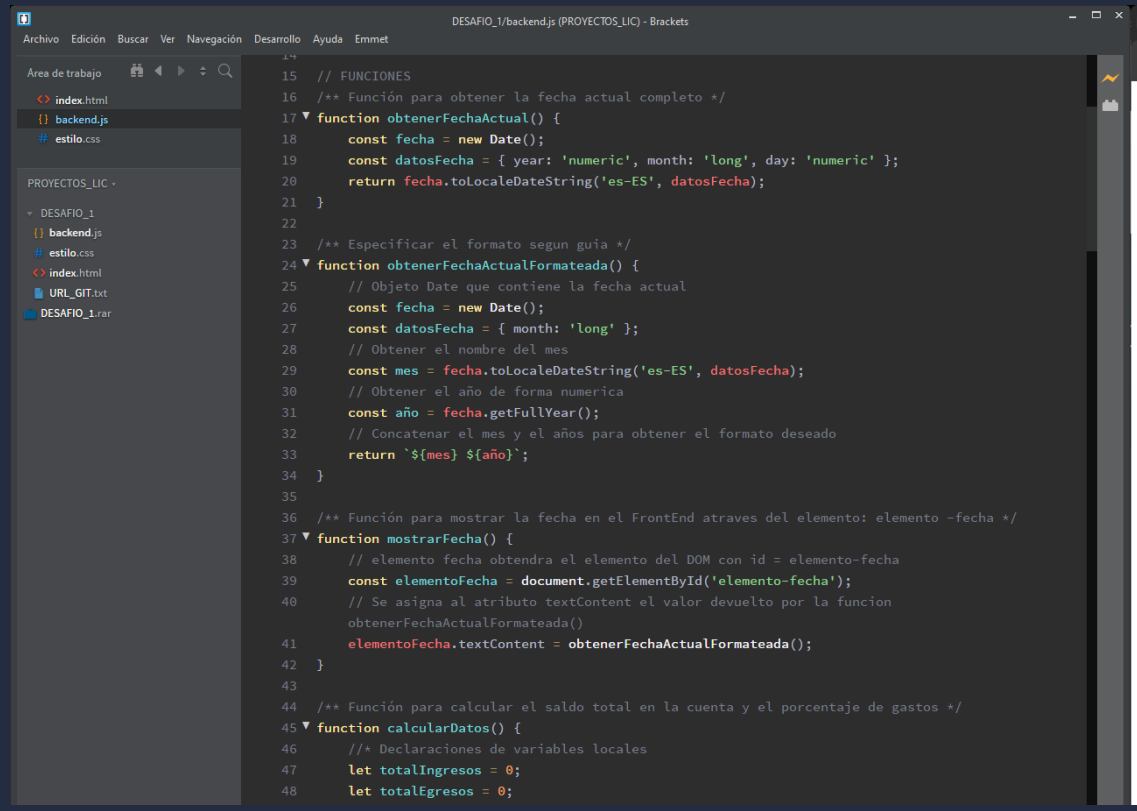
Este archivo que ayuda para diseño y es estilo a la página principal o archivo llamado index.html. Es decir estilo.css se encargar definir propiedades o características de los componentes del index.html

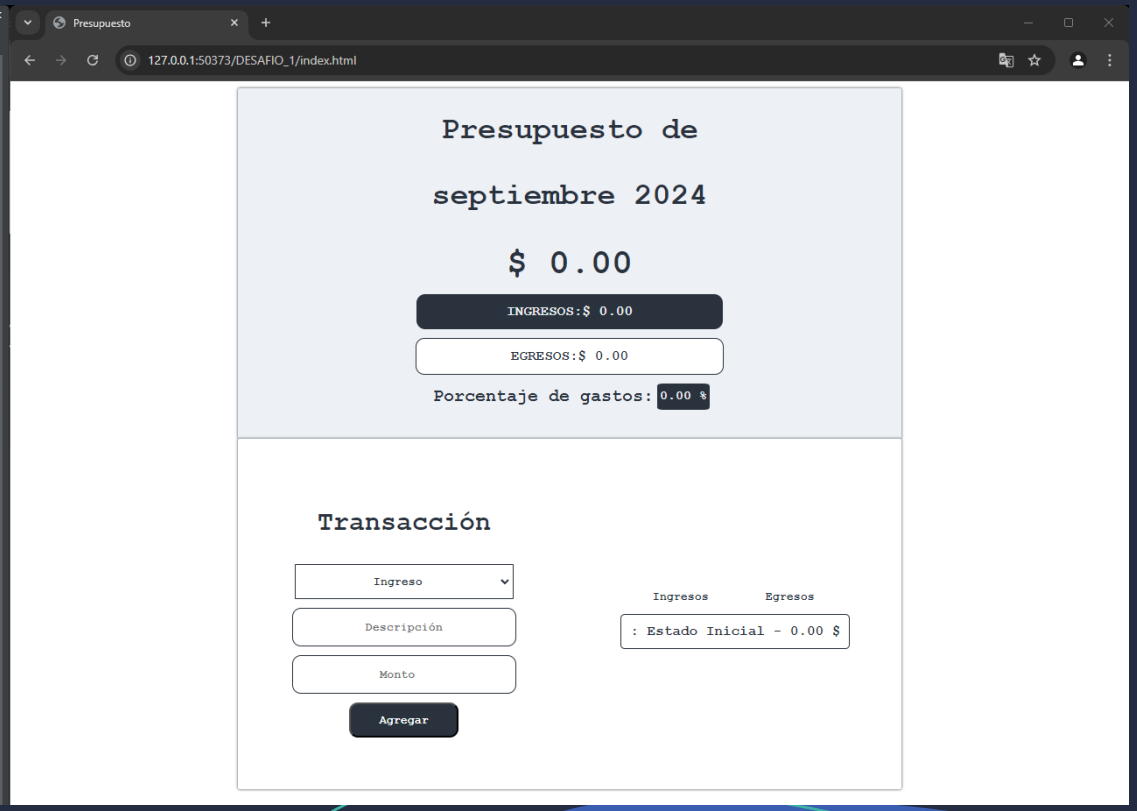
```css
.container2 {
    border-radius: 2px;
    box-shadow: 0 0 3px #2a323d;
    background-color: #ffffff;
    width: 60%;
    height: 400px;
    margin: 0 auto;
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: center;
}

.message {
    font-family: 'Courier New', Courier, monospace;
    font-size: 35px;
    font-weight: bolder;
    color: #2a323d;
    text-align: center;
    margin: 0 auto;
}
```

Herramientas de Desarrollo

IDE Brackets

Navegador : Vista Previa Dinámica





Solución de puntos del desafío

1. El mes en el título de la aplicación debe ser calculado basado en la fecha actual, por ejemplo, si la fecha actual es 23 de octubre de 2020, el título deberá ser: Presupuesto de octubre 2020.

```
<div class="message" > Presupuesto de
<p id="elemento-fecha"></p> <p class="message2" id="elemento-saldoTotal"></p>
 </div>
```

Presupuesto de

septiembre 2024

```
/** Especificar el formato segun guia */
function obtenerFechaActualFormateada() {
    // Objeto Date que contiene la fecha actual
    const fecha = new Date();
    const datosFecha = { month: 'long' };
    // Obtener el nombre del mes
    const mes = fecha.toLocaleDateString('es-ES', datosFecha);
    // Obtener el año de forma numerica
    const año = fecha.getFullYear();
    // Concatenar el mes y el años para obtener el formato deseado
    return `${mes} ${año}`;
}
```

```
/** Función para mostrar la fecha en el FrontEnd atraves del elemento: elemento -fecha */
function mostrarFecha() {
    // elemento fecha obtendra el elemento del DOM con id = elemento-fecha
    const elementoFecha = document.getElementById('elemento-fecha');
    // Se asigna al atributo textContent el valor devuelto por la funcion
    obtenerFechaActualFormateada()
    elementoFecha.textContent = obtenerFechaActualFormateada();
}
```

2. El monto total disponible en el mes deberá ser calculado con base en las transacciones generadas. La suma de los ingresos menos la suma de los egresos.

```html
<div class="message" > Presupuesto de
<p id="elemento-fecha"></p> <p class="message2" id="elemento-saldoTotal"></p>
 </div>
```

```javascript
/** Función para actualizar la interfaz con los datos calculados */
function actualizarDatos() {
    /* Cada variable contendra su valor respetando el orden de los datos
    que proviene del retorno de la lista de variales que retorna la funcion "calcularDatos()" */
    const { totalIngresos, totalEgresos, saldoTotal, porcentajeGastos } = calcularDatos();
    // Obteniendo los valores para cada uno de los elementos de DOM para mostrarlo de lado de
    cliente
    // Mostrara las cantidades en dolares y formateado a 2 decimales
    document.getElementById('elemento-fecha').textContent = obtenerFechaActualFormateada();
    document.getElementById('elemento-ingresoTotal').textContent = `$
    ${totalIngresos.toFixed(2)} `;
    document.getElementById('elemento-egresoTotal').textContent = `$
    ${totalEgresos.toFixed(2)}`;
    document.getElementById('elemento-saldoTotal').textContent = `$ ${saldoTotal.toFixed(2)}`;
    // Elemento que cotendra el valor en porcentaje de los egresos
    document.getElementById('porcentaje-gastos').textContent = `${porcentajeGastos.toFixed(2)}
    %`;
    // Se invoca a la lista de transacciones
    mostrarListaTransacciones();
}
```

Presupuesto de

septiembre 2024

$ 0.00

```javascript
/** Función para calcular el saldo total en la cuenta y el porcentaje de gastos */
function calcularDatos() {
    //* Declaraciones de variables locales
    let totalIngresos = 0;
    let totalEgresos = 0;

    // Se realiza el recorrido de una lista de transacciones guardadas temporalmente
    // Con el fin de totalizar egresos e ingresos
    transacciones.forEach(transaccion => {
        if (transaccion.tipo === 'ingreso') {
            totalIngresos += transaccion.monto;
        } else if (transaccion.tipo === 'egreso') {
            totalEgresos += transaccion.monto;
        }
    });
    // Calculo de saldoTotal
    const saldoTotal = saldoInicial + totalIngresos - totalEgresos;

    /* Validacion con condicion ternaria para evitar error de operaciones indefinida.
    Se utiliza la formula de porcentaje de egreso dada en la guia
    %egreso = (totalEgresos/ totalIngresos)*100 donde totalEgresos > 0 de lo contrario por
    defecto tendra el valor de cero.
    */
    const porcentajeGastos = totalIngresos > 0 ? (totalEgresos / totalIngresos) * 100 : 0;


    /* Una parte que me gusto aprender de JS es que se podra devolver una lista de variables sin
    necesidad de ocupar un objeto u lista como en JAVA
    */
    return {
        totalIngresos,
        totalEgresos,
        saldoTotal,
        porcentajeGastos
    };
}
```

3. Los cuadros: Ingresos y egresos, deberán mostrar las sumas de los tipos de transacciones respectivamente. En el caso del diseño, existen 2 transacciones de tipo Ingreso, lo cual suma 800.00, el monto que aparece en el cuadro de ingreso. Realizar el mismo proceso para los egresos.
    a. Para el caso de los egresos se debe calcular el porcentaje total de gastos (porcentaje de gastos) que se ha generado a partir de los egresos, tomando como ejemplo el diseño:

$$\%Egreso = \frac{TotalEgresos * 100}{TotalIngresos}$$

$$\%Egreso = \frac{300 * 100}{800}$$

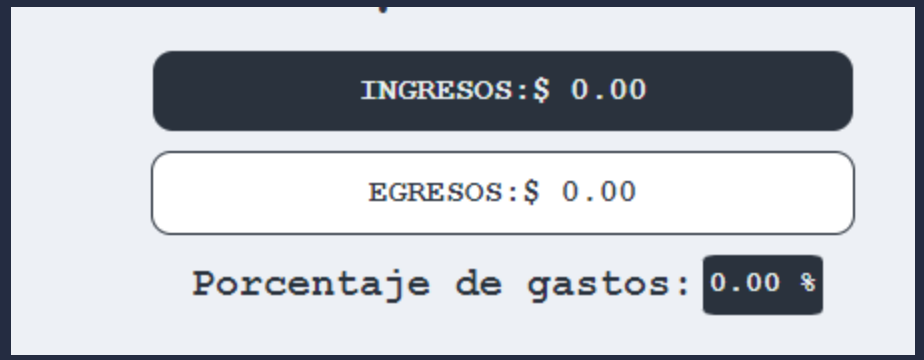
$$\%Egreso = 11.60 \sim 12\%$$

    b. Se debe utilizar la fusión ***toFixed(2)*** para realizar el redondeo de las cifras.

```
<div class="c1">INGRESOS: <p id="elemento-ingresoTotal"></p> </div>
<div class="c2">EGRESOS: <p id="elemento-egresoTotal"></p> </div>
<div class="c3">
    <div class="message3"> Porcentaje de gastos: </div>
    <div class="porcentage" id="porcentaje-gastos"></div>
```

```javascript
/** Función para calcular el saldo total en la cuenta y el porcentaje de gastos */
function calcularDatos() {
    //* Declaraciones de variables locales
    let totalIngresos = 0;
    let totalEgresos = 0;

    // Se realiza el recorrido de una lista de transacciones guardadas temporalmente
    // Con el fin de totalizar egresos e ingresos
    transacciones.forEach(transaccion => {
        if (transaccion.tipo === 'ingreso') {
            totalIngresos += transaccion.monto;
        } else if (transaccion.tipo === 'egreso') {
            totalEgresos += transaccion.monto;
        }
    });
    // Calculo de saldoTotal
    const saldoTotal = saldoInicial + totalIngresos - totalEgresos;

    /* Validacion con condicion ternaria para evitar error de operaciones indefinida.
    Se utiliza la formula de porcentaje de egreso dada en la guia
    %egreso = (totalEgresos/ totalIngresos)*100 donde totalEgresos > 0 de lo contrario por
    defecto tendra el valor de cero.
    */
    const porcentajeGastos = totalIngresos > 0 ? (totalEgresos / totalIngresos) * 100 : 0;

    /* Una parte que me gusto aprender de JS es que se podra devolver una lista de variables sin
    necesidad de ocupar un objeto u lista como en JAVA
    */
    return {
        totalIngresos,
        totalEgresos,
        saldoTotal,
        porcentajeGastos
    };
}
```

```javascript
/** Función para actualizar la interfaz con los datos calculados */
function actualizarDatos() {
    /* Cada variable contendra su valor respetando el orden de los datos
    que proviene del retorno de la lista de variales que retorna la funcion "calcularDatos()" */
    const { totalIngresos, totalEgresos, saldoTotal, porcentajeGastos } = calcularDatos();
    // Obteniendo los valores para cada uno de los elementos de DOM para mostrarlo de lado de cliente
    // Mostrara las cantidades en dolares y formateado a 2 decimales
    document.getElementById('elemento-fecha').textContent = obtenerFechaActualFormateada();
    document.getElementById('elemento-ingresoTotal').textContent = `$ ${totalIngresos.toFixed(2)} `;
    document.getElementById('elemento-egresoTotal').textContent  = `$ ${totalEgresos.toFixed(2)}`;
    document.getElementById('elemento-saldoTotal').textContent   = `$ ${saldoTotal.toFixed(2)}`;
    // Elemento que cotendra el valor en porcentaje de los egresos
    document.getElementById('porcentaje-gastos').textContent     =   `${porcentajeGastos.toFixed(2)} %`;
    // Se invoca a la lista de transacciones
    mostrarListaTransacciones();
}
```



4. En la sección de Transacción, considerar los siguientes puntos:
    a. El menú desplegable (Dropdonw) deberá incluir las opciones: Ingreso y Egreso.
    b. Agregar campo de descripción
    c. Campo para monto, que deberá ser validado para solo aceptar números.
    d. Al agregar una nueva transacción, deberá aparecer en la lista de ingresos o egresos respectivamente.

```html
<form class="inside1" id="transaction-form">
    <div class="m1">Transacción</div>
    <!-- El menú desplegable (Dropdonw) -->
    <select class="c4" id="transaction-type">
        <option value="ingreso">Ingreso</option>
        <option value="egreso">Egreso</option>
    </select>
    <input class="c5" type="text" id="description" placeholder="Descripción" />
    <input class="c6" type="number" id="amount" placeholder="Monto" step="0.01" />
    <button class="c7" type="submit">Agregar</button>
</form>
```

```
/** Manejar el envío del formulario:
    Añade un "escuchador de eventos" al formulario que se activará cuando el formulario se envíe
    (evento submit). La función que sigue será ejecutada en ese momento, y el parámetro event
    contiene información sobre el evento de envío.
*/
document.getElementById('transaction-form').addEventListener('submit', (event) => {
    // // Previene el comportamiento por defecto del formulario, se puede personalizar envio
    data
    event.preventDefault();

    const tipo = document.getElementById('transaction-type').value;
    const descripcion = document.getElementById('description').value;
    const monto = parseFloat(document.getElementById('amount').value);
    // Validar si descripcion no este vacio y que el monto sea mayor a cero
    if (descripcion && monto > 0) {
        //Añade un nuevo objeto al array a la lista de transacciones
       transacciones.push({ tipo, descripcion, monto });
        //Restablece el formulario, reinicia los valores de los campos de formulario
        document.getElementById('transaction-form').reset();
        actualizarDatos();
    }
});
```

```
/** Función para actualizar la interfaz con los datos calculados */
function actualizarDatos() {
    /* Cada variable contendra su valor respetando el orden de los datos
    que proviene del retorno de la lista de variales que retorna la funcion "calcularDatos()" */
    const { totalIngresos, totalEgresos, saldoTotal, porcentajeGastos } = calcularDatos();
    // Obteniendo los valores para cada uno de los elementos de DOM para mostrarlo de lado de
    cliente
    // Mostrara las cantidades en dolares y formateado a 2 decimales
    document.getElementById('elemento-fecha').textContent = obtenerFechaActualFormateada();
    document.getElementById('elemento-ingresoTotal').textContent = `$
    ${totalIngresos.toFixed(2)} `;
    document.getElementById('elemento-egresoTotal').textContent  = `$
    ${totalEgresos.toFixed(2)}`;
    document.getElementById('elemento-saldoTotal').textContent   = `$ ${saldoTotal.toFixed(2)}`;
    // Elemento que cotendra el valor en porcentaje de los egresos
    document.getElementById('porcentaje-gastos').textContent     =
    `${porcentajeGastos.toFixed(2)} %`;
    // Se invoca a la lista de transacciones
    mostrarListaTransacciones();
}
```

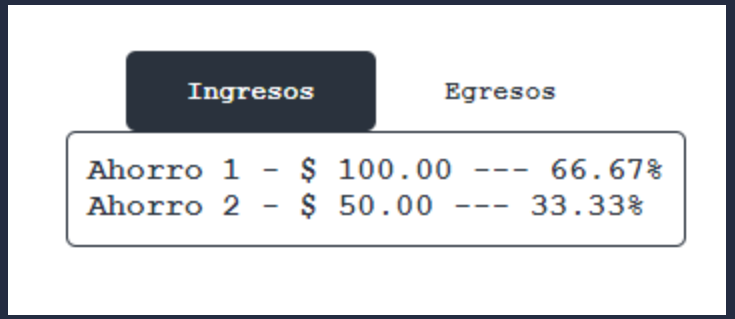
```
// Función para mostrar la lista de transacciones
function mostrarListaTransacciones() {
    const dataList = document.getElementById('data-list');
    dataList.innerHTML = ''; // Limpiar la lista
    // Recorre la lista de transacciones
    transacciones.forEach((transaccion, index) => {
        /* Se crea un nuevo elemento div y se almacena en la constante item. Este div servirá para
        contener la información de una transacción individual. */
        const item = document.createElement('div');
        //Convierte el primer carácter del tipo de la transacción a mayúscula y concatena el resto del
        tipo en minúsculas.
        item.textContent = `${transaccion.tipo.charAt(0).toUpperCase() + transaccion.tipo.slice(1)}:
        ${transaccion.descripcion} - ${transaccion.monto.toFixed(2)} $`;
        // Agrega el elemento div recién creado y configurado al final del contenido del elemento
        dataList
        dataList.appendChild(item);
    });
}
```

5. En la sección de detalles, considerar los siguientes puntos:
    a. Existirá 2 pestañas (tabs) con las opciones Ingresos y egresos, que, al hacer clic sobre alguna de ellas, deberá mostrar la lista de las transacciones realizadas.
    b. Para el caso del tab de ingresos, deberá mostrarse el texto de descripción, junto con el monto del ingreso.
    c. Para el caso del tab de egresos, deberá mostrarse el texto de descripción , junto con el monto del egreso y el calculo del porcentaje de gasto que representa la transacción

El calculo deberá ser efectuado de la siguiente forma:

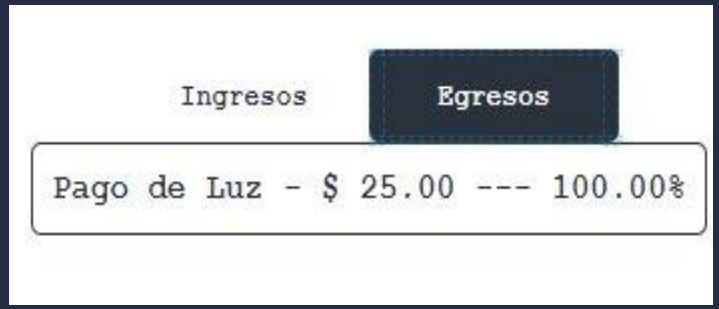
$$\%DetalleEgreso = \frac{MontoEgreso * 100}{TotalIngresos}$$

```html
    <div class="inside2">
        <div class="buttoncontainer">
            <button class="c8" id="show-incomes">Ingresos</button>
            <button class="c9" id="show-expenses">Egresos</button>
        </div>
        <div class="data" id="data-list">  </div>
    </div>
```




```javascript
/** Lista de Ingresos */
document.getElementById('show-incomes').addEventListener('click', () => {
    const dataList = document.getElementById('data-list');
    dataList.innerHTML = ''; // Limpiar la lista
 // Totaliza los montos para ser ocupados en la funcion
    const { totalIngresos, totalEgresos, saldoTotal, porcentajeGastos } = calcularDatos();

    transacciones.filter(t => t.tipo === 'ingreso').forEach((transaccion, index) => {
        const item = document.createElement('div');
        // Porcentaje por transaccion de Egresos
         const porcentaje = totalIngresos > 0 ? (transaccion.monto.toFixed(2) / totalIngresos) * 100 : 0;

        item.textContent = `${transaccion.descripcion} - $ ${transaccion.monto.toFixed(2)} ---  ${porcentaje.toFixed(2)}% `;
        dataList.appendChild(item);
    });
});
```

```html
    <div class="inside2">
        <div class="buttoncontainer">
            <button class="c8" id="show-incomes">Ingresos</button>
            <button class="c9" id="show-expenses">Egresos</button>
        </div>
        <div class="data" id="data-list">  </div>
    </div>
```



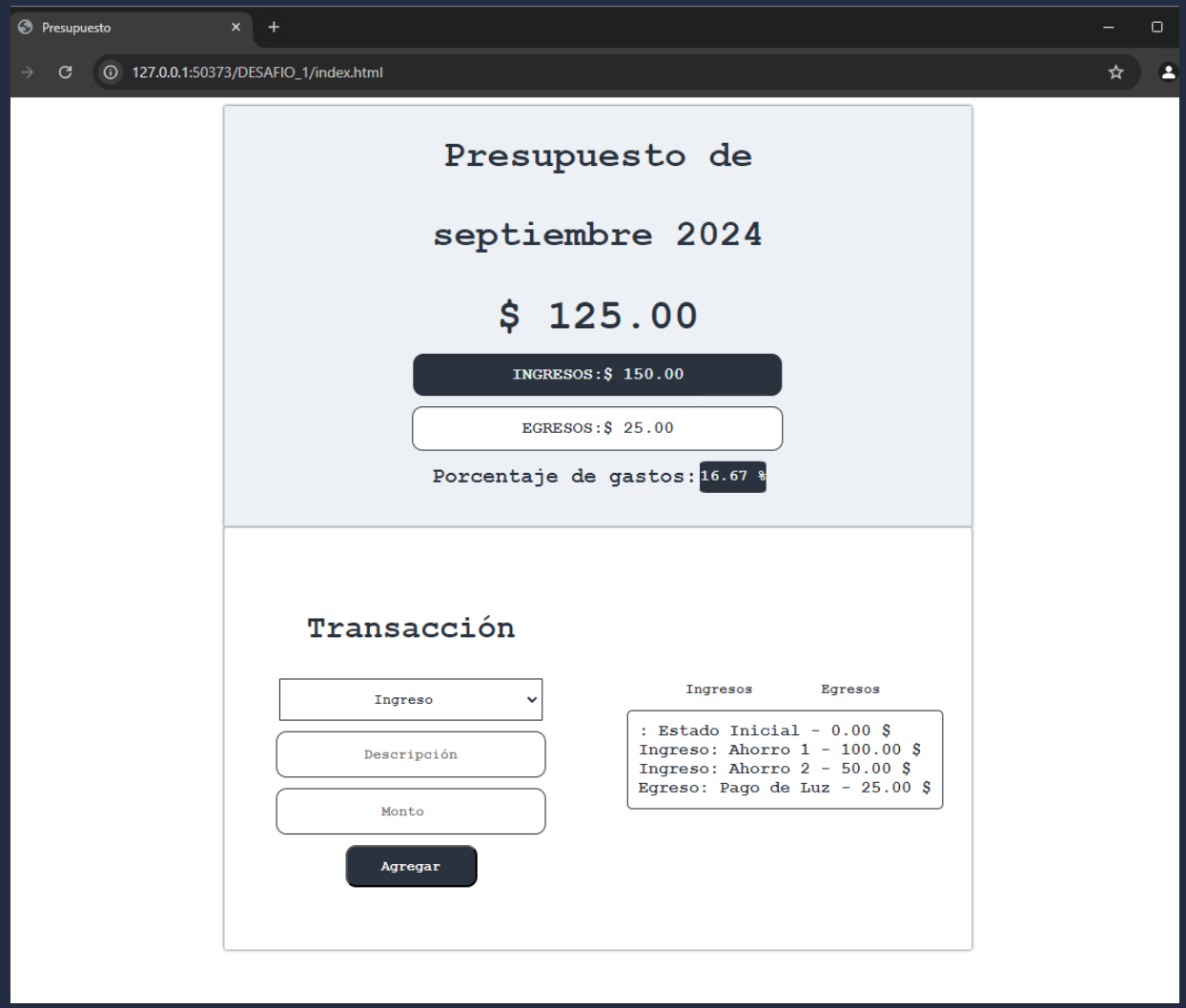

```javascript
/** Lista de Egresos */
 /* Añade un "escuchador" de eventos al elemento. En este caso, está escuchando el evento click, lo que significa que cuando el
 usuario haga clic en el elemento con ID show-incomes, se ejecutará la función proporcionada (una función de flecha en este caso).
 */
document.getElementById('show-expenses').addEventListener('click', () => {
    // Lista o contenedor que almacenara las transacciones filtradas
    const dataList = document.getElementById('data-list');
    dataList.innerHTML = ''; // Limpiar la lista
    // Totaliza los montos para ser ocupados en la funcion
    const { totalIngresos, totalEgresos, saldoTotal, porcentajeGastos } = calcularDatos();

   /* Filtra el array transacciones para obtener solo aquellas transacciones cuyo tipo es 'ingreso' */
    transacciones.filter(t => t.tipo === 'egreso').forEach((transaccion, index) => {
        // Crea un nuevo elemento div
        const item = document.createElement('div');
        // Porcentaje por transaccion de Egresos
         const porcentaje = totalEgresos > 0 ? (transaccion.monto.toFixed(2) / totalEgresos) * 100 : 0;
        // Establece el texto del nuevo elemento <div>, el monto aproximacion con dos digitos
        item.textContent = `${transaccion.descripcion} - $ ${transaccion.monto.toFixed(2)} ---  ${porcentaje.toFixed(2)}% `;
        // Agrega a la lista el nuevo div, con la informacion
        dataList.appendChild(item);
    });
});
```

```html
    <div class="inside2">
        <div class="buttoncontainer">
            <button class="c8" id="show-incomes">Ingresos</button>
            <button class="c9" id="show-expenses">Egresos</button>
        </div>
        <div class="data" id="data-list">  </div>
    </div>
...
```

```javascript
/** Aquí, se está añadiendo un "event listener" al objeto document para que ejecute una función cuando el evento DOMContentLoaded ocurra. Este evento se dispara cuando el contenido HTML del documento ha sido completamente cargado y parseado, pero antes de que se carguen las hojas de estilo, imágenes, etc */
document.addEventListener('DOMContentLoaded', () => {
    // selecciona todos los elementos en el documento que tienen las clases c8 o c9
    const buttons = document.querySelectorAll('.c8, .c9');

     /* El método querySelectorAll devuelve una lista (similar a un array) que contiene todos estos elementos. Se almacena en la constante buttons. */
    buttons.forEach(button => {
        // Se añade un "event listener" al botón actual para que ejecute una función cuando se haga clic en él.
        button.addEventListener('click', () => {

            // Quita la clase 'selected' de todos los botones
            buttons.forEach(btn => btn.classList.remove('selected'));

            // Añadir la clase 'selected' al botón clickeado
            button.classList.add('selected');
        });
    });
});

// Inicializar la interfaz al cargar la página
document.addEventListener('DOMContentLoaded', actualizarDatos);
```


jun
jul
aug
sep
oct
nov
dec
feb
mar
apr
may
jun
jul
aug
sep
oct
nov
dec
125,058
125,487
124,000
105,450
154,568
56,845
110,000
150,000
35,000
83,000
95,054
97,511
99,011
99,216
101,090
101,684

Resultados


Presupuesto
127.0.0.1:50373/DESAFIO_1/index.html
Presupuesto de
septiembre 2024
$ 125.00
INGRESOS:$ 150.00
EGRESOS:$ 25.00
Porcentaje de gastos: 16.67 %
Transacción
Ingreso
Descripción
Monto
Agregar
Ingresos
Egresos
: Estado Inicial - 0.00 $
Ingreso: Ahorro 1 - 100.00 $
Ingreso: Ahorro 2 - 50.00 $
Egreso: Pago de Luz - 25.00 $