Universidad CENFOTEC
SOMOS LO QUE SABEMOS

Indicadores Proyecto

Fundamentos de Programación Web

Profesor

Francisco Jimenez Bonilla

Estudiante

Nicole Masco Ccopa

Fecha

04/12/2025

Nombre de la Aplicación: EasyDeco

Justificación de la Aplicación:

EasyDeco nace de una molestia común: la decoración es lenta e incierta al principio.

Cuando quieres saber el costo de un proyecto de diseño o remodelación, las páginas te obligan a enviar un mensaje y esperar. Muchas veces la respuesta tarda días o nunca llega. Yo, como cualquier usuario, prefiero saber de inmediato si un proyecto es viable o no.

Esta aplicación elimina esa espera. EasyDeco permite a cualquiera generar su propio presupuesto aproximado al instante solo con las medidas y la selección de estilo.

El objetivo es hacer que el inicio de cualquier proyecto de decoración sea rápido, transparente y manejable. Es la herramienta que te da control sobre los costos sin tener que depender de la disponibilidad de un tercero.

1. **Programación JS (funciones, operadores, condicionales, ciclos, arreglos)**
   Funciones: Se cumplen a lo largo de todo el código con funciones como: calcularCotizacion(), guardarEnHistorial(), renderizarItems(), crearElemento(), descargarPDFCotizacion

   Operadores: Aritméticos (+, *, /) en la función procesarCalculo para obtener area, subtotalMateriales, manoObra y totalFinal. Lógicos (&&, `

   Condicionales: Se usan extensamente los bloques if / else y return para: Validar los inputs (largo > 0), Manejar errores (!estiloId), Filtrar en renderizarItems() (i.tipo === 'material'), y Confirmar acciones (if (resp.isConfirmed) en calcularCotizacion).

   Ciclos: Se utilizan los métodos de arreglo forEach y map para: Renderizar elementos (CATALOGO.forEach en renderizarGaleriaEstilos), Recorrer selecciones de usuario (matIds.forEach en procesarCalculo)

   Arreglos: El CATALOGO completo es un arreglo de objetos. También se usan arreglos para: Almacenar el historial (almacen), y Recolectar los ítems seleccionados (materialesSeleccionados, productosSeleccionados).

2. **Eventos y métodos de programación**
   addEventListener: Se usa en la sección "BIND eventos" para asignar las funciones a los botones principales (btnCalcular, btnGuardar, btnDescargarCotizacion, btnEnviarCorreo). También se usa en renderizarGaleriaEstilos y renderizarItems para la interacción dinámica (selección de estilos/productos).

   Métodos del DOM: La función crearElemento(tag, className) encapsula el método document.createElement(). Los métodos appendChild() y classList.add/remove/toggle se utilizan masivamente en renderizarGaleriaEstilos, renderizarItems y mostrarDesglose para construir la interfaz sin HTML en strings.

3. **Cálculos financieros**
   Función procesarCalculo: Aquí se ejecuta la lógica financiera central del proyecto:
   * Área: area = largo * ancho
   * Materiales: costo = precioPorM2 * area
   * Productos: costo = precioUnitario * cantidad
   * Mano de Obra: manoObra = MANO_OBRA_POR_M2 * area
   * Total: totalFinal = manoObra + subtotalMateriales + subtotalProductos

4. **Búsquedas**

Método find()

Se usa el método .find() en los arreglos para localizar datos:

* En seleccionarEstilo y procesarCalculo para encontrar el objeto completo del estilo dentro del CATALOGO (CATALOGO.find(g => g.estiloId === estiloId)).

* En procesarCalculo para encontrar el objeto de un material o producto específico dentro de los items del estilo, usando su id.

5. **Filtrado de datos**
   Método filter()

   Se usa el método .filter() para subdividir y organizar datos:

   * En renderizarItems, se filtran los items de un estilo por su tipo (material o producto) para mostrarlos en sus secciones correspondientes: materiales = grupo.items.filter(i => i.tipo === 'material').

6. **Controles de formulario programados**

   Control DOM

   Todos los inputs (inputLargo, inputAncho, inputTipoHabitacion, inputCorreo) son referenciados usando document.getElementById en la sección "SELECTORES DOM".

   Manipulación

   En calcularCotizacion, los valores son leídos (inputLargo.value, inputAncho.value). En restaurarHistorial, los valores de los inputs son reescritos con los datos del historial (inputLargo.value = item.largo).

7. **Validaciones con micro interacciones**

   Validaciones

   Función calcularCotizacion y validarEmail:* Se valida que largo y ancho sean valores positivos.* Se valida el formato del correo electrónico usando una Expresión Regular (validarEmail).

   Micro Interacciones

   La función **shakeElemento(el)** agrega y quita la clase CSS .shake para generar un efecto visual cuando una validación falla (ej. si el largo es inválido), lo que constituye una micro interacción visual.

   Desplegable (SweetAlert)

   Se utiliza **Swal.fire()** para mostrar alertas claras sobre errores de validación (ej. "Largo inválido", "Email inválido").

8. **Uso de programación JSON y LOCALSTORAGE**

   localStorage

   El CATALOGO completo es una estructura de datos JSON (Arreglo de Objetos) que funciona como base de datos de la aplicación.

   Se usa en guardarEnHistorial(), renderizarHistorial() y eliminarHistorial():

   * Guardar: localStorage.setItem('easydeco_historial', JSON.stringify(almacen));

   * Leer: `const almacen = JSON.parse(localStorage.getItem('easydeco_historial')

9. **Soluciones programadas usando los siguientes medios: o por pantalla o por desplegable SweetAlert o por correo electrónico**
   Medio:
   Por Pantalla (DOM)
   Función mostrarDesglose(): Esta función toma el objeto de cotización y lo renderiza línea por línea en el elemento desgloseEl, cumpliendo el requisito de mostrar el resultado en la interfaz.

   Por Desplegable (SweetAlert)
   Se utiliza Swal.fire() en **calcularCotizacion** (para mostrar el total), **guardarEnHistorial** (confirmación de guardado) y **calcularCotizacion** (para la pregunta de continuar sin ítems).

10. **Base de datos JSON**
    Estructura
    El **const CATALOGO = [...]** actúa como la base de datos de la aplicación, almacenando todos los estilos, materiales y productos con sus respectivos precios y descripciones en formato JSON (JavaScript Object Notation).