Verificación de procedimientos para la definición de componentes frontend de la aplicación (listas de chequeo)

Evidencia: GA7-220501096-AA4-EV02

Formación: Tecnología en Análisis y Desarrollo de Software

Aprendiz: Jefferson Harbey Mendez Castellanos

Instructor: John Alejandro Niño Tambo

Ficha: 2977395

Fecha: 01 – 11 – 2025

Contenido

Introducción

El desarrollo y uso de componentes Frontend de manera adecuada es fundamental para cualquier interfaz de un software, el frontend es la conexión del sistema con el usuario y es clave realizar muchas pruebas y prestar atención a cada detalle que pueda causar una mala experiencia para el usuario, o que sea incoherente con los requerimientos del proyecto. A medida que avanza el tiempo y junto con él la tecnología, van surgiendo nuevas opciones tecnológicas que nos ayudan a optimizar nuestras interfaces, reutilizando código mediante la definición de componentes reutilizables, evitando asi la duplicación de código logrando sistemas escalables y mucho más fáciles de mantener. A lo largo de este trabajo se estará perfeccionando la estructura Frontend del Software de Gestión Administrativa Matrona, y presentando un análisis detallado a cada componente que se está utilizando en la interfaz de usuario, esto con el fin de aumentar la usabilidad del sistema en la parte del frontend y optimizar el código, eliminando posible código redundante en nuestros componentes.

Objetivo

Realizar análisis detallado a los componentes frontend utilizados en la interfaz del Software de Gestión Administrativa Matrona, eliminar código redundante y justificar el uso de cada componente, asi mismo realizar lista de chequeo que nos permita identificar las fallas en la interfaz, garantizando una excelente usabilidad, y asi mismo, verificar si la interfaz está cumpliendo con los requerimientos.

Componentes

Header

```
<header class="bg-orange-100 flex justify-between items-center
p-4">
 <h2 class="text-lg font-bold text-orange-600">Panel principal
Matrona</h2>
   <div class="flex items-center space-x-4">
 <button class="btn-admin flex items-center space-x-2 bg-orange-
600 text-white px-4 py-2 rounded-lg hover:bg-orange-700
transition">
     <i class="fas fa-user"></i>
 <span id="nombreUsuario">Administrador</span>
<span id="rolUsuario" class="text-sm text-orange-200"></span>


              </button>
        </div>
 </header>
```

A continuación, se describe cada elemento que compone este importante componente presenta en la mayoría de las interfaces del sistema: **<header></header>** etiqueta HTML para definir el encabezado de cada una de las interfaces.

Clases Tailwind css para el estilo del header:

- **bg-orange-100**: equivale a el color **#ffe0b2** naranja claro el cual garantiza la coherencia de colores.
- **flex:** garantiza responsividad en pantallas pequeñas
- **justify-between**: nos agrega espacio entre los elementos como titulo y botón

- **items-center**: necesario para centrar los elementos verticalmente

<h2></h2>: Titulo que contiene el texto de la interfaz

Clases tailwind css:

- **line-height:** define un tamaño de 18px de altura al texto
- **font-bold**: negrita para el titulo
- **text-orange-600**: equivalente a #ea580c naranja fuerte al texto

** ** contenedor para dar manejo al botón administrador

Clases tailwind css utilizadas:

- **flex ítems-center**: centrar el contenido
- **space-x-4**: mover el contenido hacia la izquierda

<button></button> botón para mostrar la información del usuario en sesión actual

Clases Tailwind css

- **flex**: centrar elementos dentro del boton
- **ítems-center**: alinier contenido al centro
- **space-x-2**: 0.5rem hacia la izquierda de cada elemento hijo, en este caso el contenido dentro del botón.
- **bg-orange-600**: aplica el color naranja oscuro al botón
- **text-white**: texto en color blanco

- **px-4**: padding Izquierdo y derecho de 16px
- **py-2**: paddin top y button de 8px
- **rounded-lg**: borde a los botones
- **hover:bg-orange-700 transition:** se añade una transición para que cambie de color al pasar el cursor por encima.

"fas fa-user" elemento que contiene la clase con el icono font-awesome

****el primer elemento contiene el texto administrador y el id, y la segunda etiqueta span contiene el id del rol y la clase text-sm equivalente a 14px con un color naranja claro que posteriormente serán manejados dinámicamente por el usuario real registrado en la base de datos.

Barra Lateral de Navegación

```
<aside class="w-64 bg-orange-400 text-black font-bold border-r-2
rounded-r-xl border-orange-600 shadow-2xl shadow-black/70 z-50">
        <div class="p-4 flex items-center
center  border-b-2 border-orange-700">
            <h1  class="text-xl font-bold "><i class="fa-solid
fa-bars mr-3 ml-0 text-red-800"></i> <a href="menu.html">Menu
Principal</a></h1>
        </div>
        <nav class="p-4">
            <ul class="space-y-2">
                <li>
                    <a href="/inventario" class="flex items-
center  p-2 rounded hover:bg-orange-700 transition">
```

```
                        <i class="fas fa-box mr-3 text-red-
800"></i> Inventario
                    </a>
                </li>
                <li>
                    <a href="/materiales" class="flex items-
center p-2 rounded hover:bg-orange-700 transition">
                        <i class="fas fa-boxes mr-3 text-red-
800"></i> Materiales
                    </a>
                </li>
                <li>
                    <a href="/contabilidad" class="flex items-
center p-2 rounded hover:bg-orange-700 transition">
                        <i class="fas fa-calculator mr-3 text-
red-800"></i> Contabilidad
                    </a>
                </li>
                <li>
                    <a href="/catalogo" class="flex items-center
p-2 rounded hover:bg-orange-700 transition">
                        <i class="fas fa-wine-bottle mr-3 text-
red-800"></i> Catálogo de Productos
                    </a>
                </li>
                <li>
                    <a href="/empleados" class="flex items-
center p-2 rounded hover:bg-orange-700 transition">
                        <i class="fas fa-users mr-3 text-red-
800"></i> Empleados
```

```
                    </a>
                </li>
                <li>
                    <a href="gestionClientes.html" class="flex
items-center p-2  rounded hover:bg-orange-700 transition">
                        <i class="fas fa-user-tie mr-3 text-red-
800"></i> Clientes
                    </a>
                </li>
                <li>
                    <a href="/proveedor" class="flex items-
center p-2 rounded hover:bg-orange-700 transition">
                        <i class="fas fa-truck mr-3 text-red-
800"></i> Proveedores
                    </a>
                </li>
            </ul>
        </nav>
    </aside>
```

Aside: este componente es clave para dar forma a nuestra interfaz, presente en cada pantalla del sistema para facilitar la navegación del usuario por todos los módulos del sistema, dentro de aside se encuentra un nav que contiene cada uno de los enlaces, y así mismo, cada enlace contiene un estilo css especial para dar una excelente forma visual a cada boton. A continuación, se justifica cada elemento utilizado para crear este componente.

<aside></aside>: contiene la barra lateral, contenido necesita estilos aparte de los enlaces de navegación.

Estilos tailwind css

- **w-64:** establece un ancho de 256 px
- **bg-orange-400:** agrega el color naranja #fb923c a la barra lateral
- **text-black**: establece el texto a color negro
- **font-bold:** texto en negrita
- **border-r-2**: agrega un borde de 2px a la derecha del aside
- **border-orange-600**: agrega un color naranja oscuro a la barra
- **shadown-2xl:** agrega una sombra a la barra, esto hace obtener un estilo moderno y elegante
- **shadow-black/70**: agrega un color oscuro a la sombra
- **z-50:** establece el z-indez en 50, esto es ideal para que la barra aparezca siempre por encima de cualquier objeto en la interfaz

** ** este div contiene el boton menu, el cual se encarga de redirigir al usuario a la interfaz principal, sin importar en que lugar del sistema se encuentre, dentro del div se encuentra un h1 con el texto y la clase con el icono de font awesome

Clases tailwind css

- p-4: establece el paddin en 16px
- flex ítems-center: agregamos flex al contenedor y establecemos ítems center para centrar los elementos verticalmente

- justify-center: centramos los elementos ahora horizontalmente
- border-b-2: agregamos un borde de 2px en la parte inferior
- text-xl: definimos el tamaño del titulo equivalente a 20px
- font-bold: establecemos negrita para el titulo
- dentro de la clase del elemento i se utilizan propiedades mr-3 para definir un margen de 12px y text-red-800 para personalizar la posición y el color del icono menu

<nav></nav> se utiliza este elemento HTML para definir la sección de enlaces a los diferentes módulos del sistema

Clases Tailwind css

- **p-4**: establecemos un padding de 16px para controlar la posición del contenido

**
** elemento HTML para crear la lista no ordenada que contendrá cada uno de los enlaces

space-y-2: lo utilizamos para controlar el espacio entre los elementos en este caso li añadiendo un espaciado vertical equivalente a 8px

**
** definimos cada enlace de manera individual dentro de la lista ul

<a>incluye el href con el enlace que contiene la dirección a cada módulo del sistema

**** etiqueta con la clase que contiene cada icono font awesome

Clases Tailwind CSS

- **flex-items-center**: centramos los elementos verticalmente

- **p-2 rounded:** añadimos un padding equivalente a 8px y redondeamos los bordes para que quede idéntico a un boton
- **hover:bg-orange-700 transition**: esablecemos un hover para que al pasar el cursor por encima aparezca un color naranja oscuro convirtiendo el enlace en botón de forma más intuitiva para el usuario
- **fas fa**: contiene el icono que incluimos desde font awesome
- **mr-3**: añadimos un margen derecho para separar el icono del texto
- **text-red-800**: se agrega un color #991b1b a los iconos, equivalente a un rojo oscuro para que contraste con el resto de los elementos de la barra
- finalmente, el resto de los enlaces contiene los mismos estilos los cuales los convierten en un botón elegante, completando así nuestra barra lateral.

Footer

```
<footer class="bg-orange-100 border-t p-4 flex justify-end">
            <button id="logoutBtn" class="flex  space-x-2 bg-
orange-600 text-white px-4 py-2 rounded-lg hover:bg-orange-700
transition">
                <a href="/auth/login" class="flex items-center">
                    <i class="fas fa-right-from-bracket mr-
2"></i>
                    <span >Salir</span>
                </a>
            </button>
        </footer>
```

<footer></footer> componente importante y presente en todas las interfaces, su principal función es establecer el pie de página y mostrar el botón de salir el cual llevara al inicio de sesión directamente desde cualquier interfaz. Elementos que conforman este componente.

Clases tailwind css

- **flex justify-end:** añadimos flex al contenedor y lo centramos al final de la fila horizontalmente, moviedo de esta manera el botón al final del contenedor
- **bg-orange-100**: establecemos el color similar al del resto de la interfaz
- **border-t**: se añade un borde superior, decorativo
- **p-4**: establecemos un padding de 16px

<button></button>botón salir del sistema, dentro tenemos el id, el enlace **<a><a/>** el elemento **** con el icono y un **** con el texto

Clases tailwind css

- **flex:** para hacer flexible el botón a diferentes pantallas
- **space-x-2:** añadimos un espacio horizontalmente para separar elementos
- **bg-orange-600**: color naranja al botón
- **text-white:** agregamos el color blanco al texto
- **px-4**: padding izquierdo y derecho de 16px
- **py-2**: padding superior y inferior equivalente a 8px, necesario para darle la posición correcta al boton dentro del footer. Y

- **rounded-lg:** añadimos un border radios de 2px al boton
- **hover:bg-orange-700 transition:** añadimos un hover con un color naranja mas oscuro para que al pasar el cursor sobre el botón, este resalte.

Tabla de pedidos creada con divs, y renderizada con JavaScript utilizando los datos reales de cada pedido mediante una solicitud get a nuestra api en el backend que nos trae los datos desde la base de datos.

```
function renderPedido(pedido) {
    const cont = document.querySelector(".contenedor_pedidos");


    const div = document.createElement("div");
    div.className = "flex bg-orange-200 shadow-xl p-3 rounded
border-l-4 border-orange-500 mb-2";
   //los operadores ?. me sirven para evitar errores en caso de
que algun campo venga null
    div.innerHTML = `
        <div class="w-1/4">${pedido.cliente?.usuario.nombre ||
"Cliente desconocido"} ${pedido.cliente?.usuario.apellido ||
""}</div>
        <div class="w-
1/4">${pedido.detalles?.[0]?.cantidad_pedido_uds || 0}</div>
        <div class="w-1/4">${pedido.detalles?.[0]?.nombre_bebida
|| "Sin nombre"}</div>
        <div class="w-1/4">${pedido.detalles?.[0]?.presentacion
|| "N/A"}</div>
        <div class="w-1/4 text-right font-semibold
${pedido.estado === "entregado" ? "text-green-600" : "text-red-
600"}">
            ${pedido.estado === "pendiente"
```

```
                ? `<button id="btn-pedido-${pedido.id_pedidos}"
class="bg-orange-400 text-black rounded-xl p-
1">Entregar</button> Pendiente`
                : "Entregado"}
        </div>
    `;


    cont.prepend(div);
```

Formularios

Registro de usuario en el sistema

Este formulario nos permite el registro de cada usuario en el sistema, este componente lleva su estilo moderno y responsive y es la entrada de los datos claves de los usuarios, incluido el rol, a continuación, se especifica cada elemento HTML y las principales propiedades css utilizadas para construir este importante componente.

<form><form/> element HTML para definir el formulario que representa un campo de entrada de datos a nuestro sistema.

** ** necesario para controlar el estilo de cada input del formulario

<label></label> se utiliza para dar un nombre descriptivo al campo, y se usa for para definir el id del campo, en este caso contiene un id descriptivo de acuerdo al elemento que se espera de entrada.

<input><input>entrada de datos en el formulario, ya sea type text o password

<select></select> usamos este elemento para definir la sección de opciones para que el usuario seleccione el rol

<option></option> opciones para seleccionar cada rol, con valores 1, 2, 3, numero para identificar cada rol

<button></button> botón tipo submit para registrarse y botón type button para salir

Clases Tailwind css

- **flex:** necesario para hacer responsive el formulario
- **ítems-center:** centrar el formulario verticalmente
- **justify-center**: centrar el formulario horizontalmente
- **bg-orange-100, bg-orange-200:** para obtener el contraste correcto y diferenciar el componente con el formulario del color del fondo de pantalla
- **text-gray:** para todo el texto del formulario
- entre otras clases tailwind decorativas como **focus:ring-orange** para colorear los bordes en naranja al hacer click en el input

Formulario para agregar nuevas cervezas al inventario

```
<form>
                            <div class="grid grid-cols-1
md:grid-cols-2 gap-6">
                                <!-- part1 -->
                                <div class="space-y-4">
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Nombre Cerveza</label>
                                        <input
id="nombre_cerveza" type="text" class="w-full px-3 py-2 border
border-gray-300 rounded-md focus:outline-none focus:ring-2
focus:ring-orange-500" placeholder="Ingrese nombre de cerveza">
                                    </div>
                                     <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Descripción</label>
```

```
                                        <textarea
id="descripcion" class="w-full px-3 py-2 border border-gray-300
rounded-md focus:outline-none focus:ring-2 focus:ring-orange-
500" rows="2" placeholder="Descripción de la
cerveza"></textarea>
                                    </div>
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Contenido (ml)</label>
                                        <input  id="contenido"
type="number" class="w-full px-3 py-2 border border-gray-300
rounded-md focus:outline-none focus:ring-2 focus:ring-orange-
500" placeholder="Contenido de cerveza">
                                    </div>
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Porcentaje de
Alcohol</label>
                                        <input id="alcohol"
type="text" class="w-full px-3 py-2 border border-gray-300
rounded-md focus:outline-none focus:ring-2 focus:ring-orange-
500" placeholder=" Ingrese % vol alcohol">
                                    </div>
                                </div>
                                <div class="space-y-4">
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Stock Inicial</label>
                                        <input
id="stock_inicial" type="number" class="w-full px-3 py-2 border
border-gray-300 rounded-md focus:outline-none focus:ring-2
focus:ring-orange-500" placeholder="Unidades listas para
vender">
```

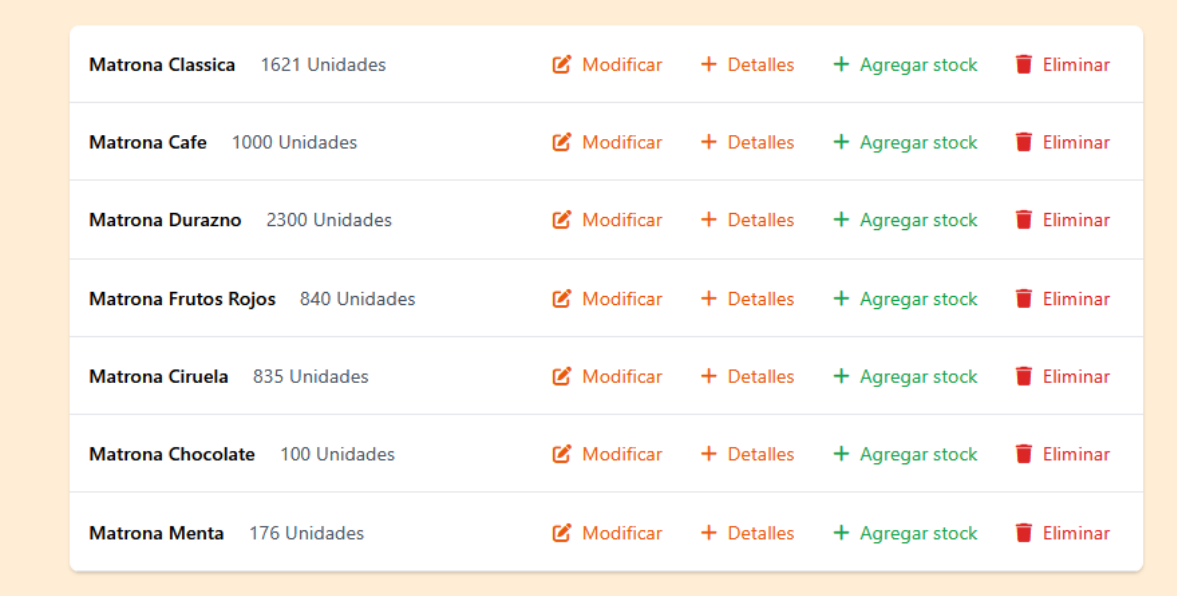
```
                                    </div>
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Precio por
Unidad</label>
                                        <input
id="precio_unidad" type="text" class="w-full px-3 py-2 border
border-gray-300 rounded-md focus:outline-none focus:ring-2
focus:ring-orange-500" placeholder="Ingrese precio unidad">
                                    </div>

                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Precio Sixpack</label>
                                        <input
id="precio_sixpack" type="text" class="w-full px-3 py-2 border
border-gray-300 rounded-md focus:outline-none focus:ring-2
focus:ring-orange-500" placeholder="Ingrese precio sixpack">
                                    </div>
                                    <div>
                                        <label class="block
text-sm font-medium text-gray-700 mb-1">Precio Caja</label>
                                        <input id="precio_caja"
type="text" class="w-full px-3 py-2 border border-gray-300
rounded-md focus:outline-none focus:ring-2 focus:ring-orange-
500" placeholder="Ingrese precio caja">
                                    </div>
                                </div>
                            </div>

                            <div class="mt-8 flex justify-end
space-x-4">
```

```
                                <button id="btnGuardarProducto"
type="button" class="px-4 py-2 bg-orange-600 text-white rounded-
lg hover:bg-orange-700 transition">
                                    <i class="fas fa-save mr-
2"></i> Guardar Producto
                                </button>
                            </div>
                        </form>
```

HTML : nuevamente se hace uso de etiquetas HTML para diseñar la estructura del componente, como form, div, label, input, button entre otras etiquetas, ademas type para definir el tipo de input como text, buton y placeholder o texto descriptivo para ayudar al usuario.

CSS: se utilizan múltiples clases tailwind para dar estilo al componente, y que este sea coherente con el estilo que presenta cada interfaz del sistema, colores naranja claro, oscuro, y blanco, estilos para el texto en color naranja y gris, otras propiedades css importantes como padding, width, height entre otras, necesarias para dar estilo coherente.

Contendedor de productos/inventario



Cada fila contiene las unidades disponibles de cada cerveza, junto con sus opciones modificar, detalles agregar stock y eliminar, A continuación, se presenta la justificación de cada elemento necesario para construir este componente el cual esta renderizado para mostrar dinámicamente los datos que traemos directamente desde la base de datos.

```
async function cargarInventario() {
    try {
        const response = await
fetch("http://127.0.0.1:8000/catalogo/");
        const inventario = await response.json();

  const contenedor =
document.getElementById("productosInventario");
        contenedor.innerHTML = "";
```

```
        inventario.forEach(item => {
            console.log("Item recibido:", item);
            const productoHTML = `
                <div class="border-b p-4 hover:bg-gray-50">
                    <div class="flex items-center justify-
between">
                        <div>
                            <span class="font-
semibold">${item.inventario.nombre_bebida}</span>
                            <span class="ml-4 text-gray-
600">${item.inventario.cantidad_disponible} Unidades</span>
                        </div>
                        <div class="flex gap-2">
                            <input type="number" class="w-16
border px-2 py-1 rounded hidden" min="1" placeholder="0">

                            <a href="/editar-
inventario/${item.id_catalogo}"
                       class="text-orange-600 hover:text-orange-
800 px-3 py-1 rounded inline-block">
                       <i class="fas fa-edit mr-1"></i>
Modificar
                    </a>
                    <button class="btn-detalles text-orange-600
hover:text-orange-800 px-3 py-1 rounded"
                            data-id="${item.id_inventario}">
                    <i class="fas fa-plus mr-1"></i> Detalles
                    </button>
```

```
                    <button
onclick="abrirModal(${item.id_inventario})"
                        class="text-green-600 hover:text-green-
800 px-3 py-1 rounded">
                        <i class="fas fa-plus mr-1"></i> Agregar
stock
                    </button>
                    <button
onclick="eliminarProducto(${item.id_catalogo})"
                        class="text-red-600 hover:text-red-800
px-3 py-1 rounded">
                        <i class="fas fa-trash mr-1"></i>
Eliminar
                        </button>
                        </div>
                    </div>
                </div>
            `;
            contenedor.insertAdjacentHTML("beforeend",
productoHTML);
        });
```

A continuación se hace descripción de los elementos utilizados

** **necesarios para tener el control total de cada elemento del componente

****dentro del span insertamos dinámicamente los datos de cada cerveza registrada en la base de datos, como nombre, cantidad disponible etc.

<input></input>usado para modal que se abre dinámicamente, el cual permite agregar cervezas al stock

**** contienen los iconos

Clases tailwind css principales

Se utilizan propiedades css como flex para diseño responsive, text Orange claro y oscuro fuente semi bold y otras propiedades claves para dar estilo como margin padding, border radius entre otros

JavaScript para insertar datos dinámicamente desde la base de datos mediante la api backend

- **async function**: función asíncrona que parece síncrona para que se ejecute el bloque de código sin bloquear el hilo principal de ejecución
- **await:** para que la llamada a la api se complete antes de continuar la ejecución del resto de código
- **try catch**: necesario para manejar errores en tiempo de ejecución
- **fetch:** clave para hacer la solicitud HTTP de manera asíncrona a nuestro backend mediante nuestra api
- **${item.inventario.nombre_bebida}** utilizamos plantillas literales para asignar cada dato que viene desde el backend en JSON y convertido a Objeto al campo correspondiente.
- .**insertAdjacentHTML("beforeend")** para insertar el nuevo componente dinámicamente al final de la fila de cervezas

Flujo de datos

Se realiza la solicitud get mediante la api con fetch, los datos llegan en JSON posteriormente se transforman esos datos a un Objeto js, se crean variables para almacenar cada valor, se obtiene el componente en el HTML mediante el id, se crea un bucle para que por cada ítem recibido se cree un nuevo componente con cada uno de los datos como nombre de la cerveza, cantidad disponible y demás opciones, se agregan los datos dinámicamente de acuerdo a lo que nos llega del backend y finalmente al componente contenedor se le van insertando cada uno de los componentes que se van creando al final de la lista.

Tarjeta Bebidas en Catalogo

Componente importante para mostrar a los clientes de la empresa cada bebida disponible, cada tarjeta contiene los datos de la cerveza, y también la opcion de reservar el pedido según su

necesidad. A continuación se realiza breve resumen de elementos HTML utilizados para crear la tarjeta, y los estilos CSS utilizados para crear esta tarjeta.

HTML: la estructura de la tarjeta esta compuesta principalmente por divs “contenedores”, necesarios para organizar cada sección y aplicar estilos individuales, sea actualmente o a futuro, también esta compuesta por otros elementos HTML claves como: etiquetas h1, span, img, h4, section, option, button.

CSS: mescla de colores naranja claro, oscuro y blanco, flex y propiedades flex de acuerdo a la necesidad de la tarjeta para lograr adaptabilidad y lograr que se responsive, otras propiedades css utilizadas como: margin, padding, border radius, font bold y semibold entre otras propiedades todas necesarias para dar un estilo moderno a la tarjeta.

JavaScript: la tarjeta esta renderizada con el contenido de datos de cada cerveza, que se obtiene de la base de datos el cual llega en la api por medio de fetch, y se va insertando dinámicamente en cada campo de la tarjeta, también se usan funciones JS como toLocalString para dar formato de peso colombiano a los precios de los productos.

Tabla contabilidad

Historial de Ventas

ID PEDIDO	BEBIDA	CANTIDAD	FECHA DE PEDIDO	NOMBRE DE CLIENTE	TOTAL DE VENTA
75	Matrona Menta	6 unidades	5/10/2025	Otoniel Carvajal	$ 12.000
76	Matrona Menta	2 unidades	10/10/2025	Otoniel Carvajal	$ 4.000
77	Matrona Menta	36 unidades	10/10/2025	Agustin Gelvez	$ 72.000
78	Matrona Frutos Rojos	10 unidades	10/10/2025	Agustin Gelvez	$ 80.000

Esta tabla es útil para mostrar el historial de ventas, este historial tiene datos calves para la información contable necesaria para el cliente, de acuerdo a los requerimientos, se presenta en una estructura moderna y responsive, adaptable a todo tipo de pantalla. A continuación, se describen los principales elementos necesarios para construir este componente.

HTML

<table></table>: elemento el cual contiene todos los elementos de la tabla

<thead></thead> definimos los encabezados de la table

<tr></tr> definimos la fila de celdas

<th></th> definimos estas celdas como encabezados de la tabla

<tbody></tbody> usado para encapsular cada lista de elementos td

<td></td> definimos las celdas que van a contender los datos

Tailwind css

Se hace uso de importantes propiedades css como **overflow** para dibujar la barra de desplazamiento, y **max-heigth** de 256 pixeles para que no ocupe mucho espacio en la pantalla, se utilizan los colores habituales y demás propiedades para dar estilo elegante.

JS: datos dinámicos de acuerdo a los datos de nuestro api, la cual trae desde el backend los datos obtenidos de las query a la tabla pedidos y detalle_pedido.

```
function renderFilaPedido(p) {
    const tr = document.createElement("tr");
    tr.className = "hover:bg-gray-50";
    tr.innerHTML = `
```

```
      <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-
900">${p.id_pedido ?? p.id_pedidos}</td>

      <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-
500">${p.nombre_bebida}</td>

      <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-
500">${p.cantidad} unidades</td>

      <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-
500">${formatDate(p.fecha_pedido)}</td>

      <td class="px-6 py-4 whitespace-nowrap text-sm text-gray-
500">${p.nombre_cliente}</td>

      <td class="px-6 py-4 whitespace-nowrap text-sm font-
semibold text-green-600">${formatCurrency(p.total_venta)}</td>
    `;

    tbody.appendChild(tr);
  }
```

Otros componentes claves del sistema

Grafica de ventas en contabilidad: necesario para mostrar la herramienta para análisis de datos,

Contenedores con el total del presupuesto: muestra las unidades vendidas y el total del presupuesto

Tarjeta Empleados: tarjeta con los datos de cada empleado de la empresa

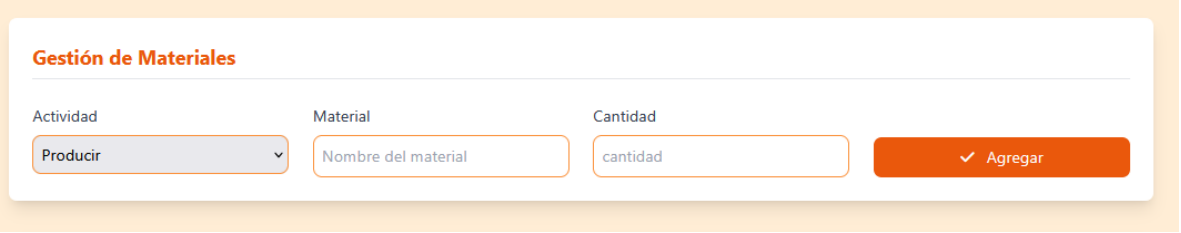
Formulario para agregar proveedores

Tabla Proveedores: necesaria para mostrar todos los proveedores

Listado de proveedores

Nombre Proveedor	Material que provee	Cantidad de Material	Frecuencia de Entrega	Numero de Telefono	Dirección
Juan Carlos	Trigo	20 Kg	Mensual	3208300967	Cerrito Santander
Carlos	Cebada	500 kg	Mensual	3201929924	Cerinza Boyaca

Form para agregar materiales: necesario para agregar materiales de producción



Contenedor con materiales: muestra al usuario los materiales que se encuentran registrados.

Listas de chequeo para validación de componentes Frontend

Aspecto a Inspeccionar	Cumple	No Cumple	Observaciones
Cada componente cumple con la funcionalidad descrita en el documento de requerimientos	✓		Los componentes frontend están diseñados de acuerdo a los requisitos
Las acciones del usuario (clics, formularios, botones, menús) producen la respuesta esperada	✓		Se han realizado múltiples pruebas positivas
Los componentes muestran correctamente los datos provenientes del backend (FastAPI / Base de datos MySQL)	✓		La conexión con el backend está funcionando correctamente
El flujo entre pantallas (navegación) es coherente con los casos de uso definidos	✓		La navegación es correcta
Los estados de carga, éxito y error están correctamente manejados	✓		Manejo de errores en el código implementado
Las validaciones de formularios funcionan	✓		Pruebas finalizaron positivamente
Los componentes respetan la guía de estilo del proyecto (colores, tipografía, espaciado)	✓		Los colores en todo el sistema son totalmente coherentes
Los botones, inputs y tarjetas mantienen consistencia visual entre pantallas	✓		Se puede evidenciar una consistencia visual entre componentes
La interfaz es intuitiva y no requiere explicaciones adicionales para el usuario	✓		La interfaz es muy sencilla de aprender a usar
Las etiquetas y textos son claros, sin errores ortográficos ni ambigüedades	✓		Textos claros
El diseño es responsive	✓		El sistema es muy adaptable a cualquier pantalla

El código es legible, con nombres descriptivos en funciones		✓	Se requiere revisión a algunos nombres de variables poco claros que causan confusión
Los componentes se cargan rápidamente sin bloqueos visuales	✓		Hasta el momento no se evidencian bloqueos visuales
Las imágenes y recursos están optimizados	✓		Imágenes guardadas en cloudinary.
Los colores tienen suficiente contraste.	✓		Contraste exelente
Cada componente fue probado en los principales navegadores (Chrome, Firefox, Edge).		✓	Positivo en Chrome, Firefox, faltan pruebas en mas navegadores
No hay errores visibles en consola del navegador	✓		La consola no muestra errores
Se incluyen botones hacia los módulos requeridos por el cliente facilitando accesibilidad a funciones	✓		El sistema es fácil de navegar debido a su barra lateral siempre activa
¿Las peticiones a la API (GET, POST, PUT, DELETE) funcionan correctamente al realizarlas desde el frontend?	✓		Se trabaja constantemente cada vez que surge un bug para solucionarlo

Conclusiones

Se ha realizado un trabajo completo, revisando cada uno de los principales componentes Fronted utilizados en nuestro Software de Gestión Administrativa Matrona, se especificó y se justificó su uso y se revisó detalladamente cada elemento HTML y estilos CSS utilizados para dar forma a cada componente de nuestra interfaz, así mismo, se revisó el uso de datos dinámicos, los cuales se obtienen directamente desde la base de datos mediante la gestión de nuestra api la cual nos hace el procedimiento en el backend y nos trae los datos reales para insertarlos en el Frontend, logrando así la conexión completa del sistema y por tanto un software increíblemente funcional. Durante la revisión se han detectado muchos errores, se ha eliminado código duplicado y se trabaja en lograr la reutilización de código, de esta manera se puede lograr un software mas mantenible a largo plazo, por ultimo se realizó una lista de chequeo en la cual se inspecciono en múltiples aspectos, detectando nuevamente varios errores e inconsistencias en el sistema de manera oportuna. Finalmente podemos concluir que el frontend de nuestro software responde adecuadamente a los requerimientos del cliente, su respuesta es muy rápida, presenta un diseño responsive, lo cual lo hace adaptable a pantallas mas pequeñas sin romper el diseño, aún tiene código repetitivo por lo cual es necesario trabajar en opciones modernas que nos permitan reutilizar componentes logrando mucho más la optimización el código.