UNIVERSIDAD DEL VALLE

MANUAL TÉCNICO FRONTEND

"PROYECTO FINAL EMPRENDEHUB"

NOMBRE: Juan Marcos Sirpa Canqui

DOCENTE: Henry Miranda Ordonez

MATERIA: Programacion Web

FECHA**:** 08/12/2025

GRUPO: "C"

La Paz-Bolivia
2025

Tabla de contenido

1. Introducción

Este manual técnico describe de manera detallada la **estructura, funcionamiento y mantenimiento del frontend** de la página web **EmprendeHub**, un proyecto orientado a facilitar la interacción de los usuarios con información sobre categorías de productos y distribuidores en línea.

El documento está dirigido principalmente a **desarrolladores, diseñadores web o personal técnico** que necesiten comprender a fondo cómo está construido el frontend, cómo realizar modificaciones, implementar nuevas funcionalidades o mantener la página de manera eficiente. Asimismo, sirve como referencia para quienes necesiten familiarizarse rápidamente con la arquitectura del proyecto y sus principales componentes.

El **frontend** de EmprendeHub está diseñado para proporcionar una **interfaz visual intuitiva**, con navegación clara, animaciones atractivas y responsivas que mejoran la experiencia del usuario. La página hace uso de **tecnologías web estándar**, tales como **HTML5, CSS3 y JavaScript**, asegurando compatibilidad con la mayoría de los navegadores modernos y dispositivos móviles. La interacción del usuario con la página incluye elementos como botones, carruseles de productos, menús desplegables y formularios, todos implementados de manera que sean **fáciles de usar y mantener**.

Además, este manual proporciona la base para futuras **mejoras y extensiones**, como la incorporación de nuevas páginas, secciones interactivas o cambios en la estética visual, manteniendo la coherencia con la arquitectura existente.

2. Objetivo del Manual

El principal propósito de este manual es servir como una guía técnica completa para el **desarrollo y mantenimiento del frontend** de EmprendeHub. Los objetivos específicos incluyen:

1. **Documentar la arquitectura del frontend:**
    - Presentar de manera organizada la estructura de carpetas y archivos.
    - Identificar la función de cada componente y archivo principal.
2. **Explicar el funcionamiento de scripts, estilos y componentes:**
    - Describir las funciones JavaScript que controlan la interacción del usuario.
    - Detallar el uso de estilos CSS y animaciones aplicadas a los elementos de la página.
    - Mostrar cómo se integran librerías externas como **Google Fonts, iconos y frameworks opcionales**.

3. **Facilitar el mantenimiento y futuras modificaciones del proyecto:**
    - Proveer pautas claras para agregar nuevas páginas, elementos o funcionalidades.
    - Indicar buenas prácticas de programación, estructura y nomenclatura.
    - Permitir una rápida resolución de errores o ajustes en el frontend.
4. **Servir como referencia para nuevos desarrolladores:**
    - Acelerar la curva de aprendizaje sobre la estructura y lógica del frontend.
    - Minimizar errores durante la implementación de cambios o mejoras.

3. Alcance del Frontend

Este manual se enfoca exclusivamente en la **parte visual e interactiva** de la página web, conocida como frontend, abarcando los siguientes aspectos:

- **Estructura de páginas HTML:**
    - Organización de las páginas principales (index.html, Categoria.html, Distribuidoras.html).
    - Uso de etiquetas semánticas para mejorar la accesibilidad y SEO.
- **Estilos CSS y animaciones:**
    - Implementación de estilos generales y específicos para cada sección.
    - Uso de **Flexbox y Grid** para diseño responsivo.
    - Aplicación de animaciones CSS para mejorar la experiencia del usuario.
    - Definición de colores, tipografía, márgenes, paddings y pseudo-elementos.
- **Scripts JavaScript para interacción y funcionalidad:**
    - Manejo de eventos (clicks, hover, scroll).
    - Validaciones de formularios, navegación dinámica y carruseles.
    - Modularidad del código para facilitar la extensión futura.
- **Integración con librerías externas:**
    - **Google Fonts** para tipografía personalizada.
    - Iconos y recursos gráficos optimizados para web.

Limitaciones:
Este manual **no cubre el backend**, bases de datos ni la lógica de servidor. Tampoco detalla la seguridad o gestión de usuarios en el lado del servidor, ya que su foco está en el **frontend** y la experiencia visual del usuario.

4. Requisitos Técnicos

Para desarrollar, probar y mantener el frontend de EmprendeHub, se requieren los siguientes elementos:

4.1 Navegadores compatibles

- **Google Chrome:** versión reciente
- **Mozilla Firefox:** versión reciente
- **Microsoft Edge:** versión reciente

Se recomienda probar las páginas en **múltiples navegadores y dispositivos** para garantizar consistencia en diseño y funcionalidad.

4.2 Tecnologías utilizadas

- **HTML5:** estructura semántica de páginas, formularios, enlaces, imágenes y secciones.
- **CSS3:**
    - Flexbox y Grid para maquetación responsiva.
    - Animaciones, pseudo-clases (:hover, :focus) y pseudo-elementos (::before, ::after).
    - Variables CSS y transiciones suaves para interacción visual.
- **JavaScript ES6+:**
    - Funciones y eventos para interacción del usuario.
    - Carruseles, menús dinámicos y validaciones de formularios.
    - Manipulación del DOM para actualizar contenido dinámicamente.

4.3 Entorno de desarrollo recomendado

- **Visual Studio Code:** editor de código con soporte para HTML, CSS y JS.
- **Node.js (opcional):** para dependencias, automatización y testing de scripts.
- **Navegador web:** para pruebas y depuración de frontend.

4.4 Recomendaciones adicionales

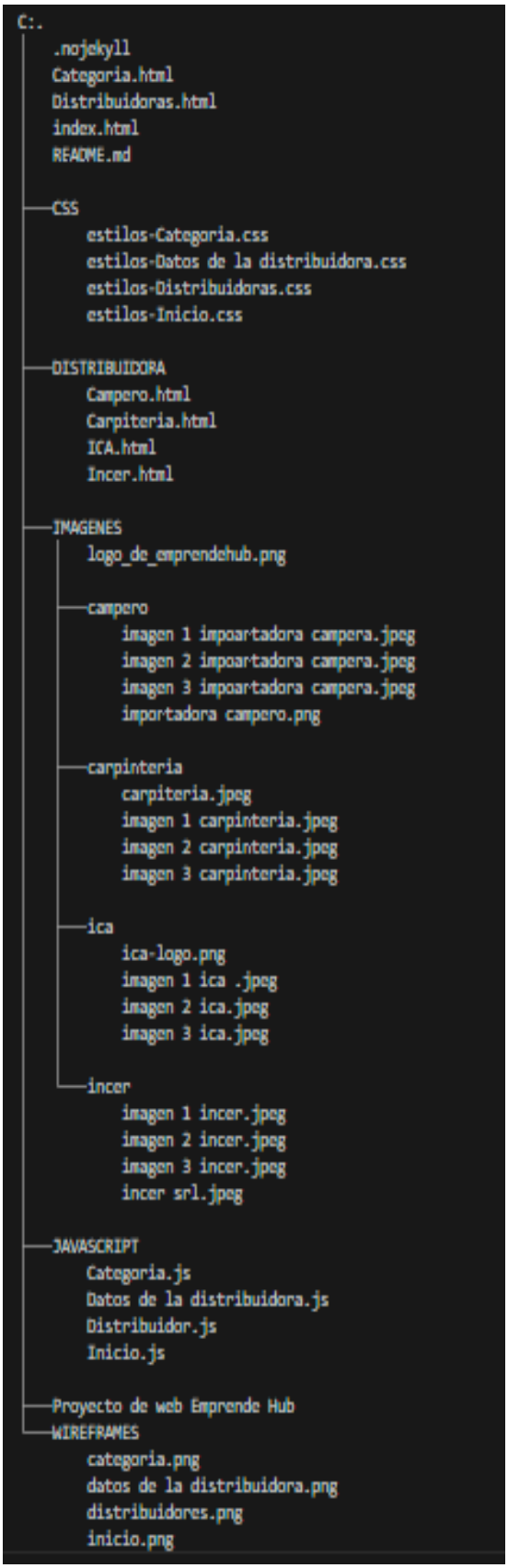
- Mantener actualizadas las librerías externas.

- Usar herramientas de desarrollo (DevTools) para depuración y análisis de rendimiento.
- Seguir buenas prácticas de nomenclatura y organización de archivos.

5. Arquitectura del Frontend

La arquitectura del frontend de la página web **EmprendeHub** está organizada de manera modular y jerárquica, facilitando la **mantenibilidad, escalabilidad y comprensión del proyecto**. La estructura de carpetas se diseñó para separar de manera clara los **archivos HTML, CSS, JavaScript, imágenes y recursos adicionales**, optimizando el flujo de trabajo y la colaboración entre desarrolladores.

5.1 Estructura de Carpetas y Archivos



5.2 Descripción de Carpetas y Archivos

Raíz del Proyecto

- .nojekyll: Archivo utilizado para habilitar GitHub Pages evitando que se interpreten archivos con guiones bajos como archivos Jekyll.
- index.html: Página principal de EmprendeHub, que sirve como punto de entrada del usuario.
- Categoria.html: Página que muestra las categorías de productos disponibles.
- Distribuidoras.html: Página que lista las distribuidoras registradas con sus datos.
- README.md: Documentación general del proyecto, información del repositorio y guía inicial.

CSS

- Cada archivo CSS está **asociado a una página específica**, permitiendo un diseño modular y fácil de mantener:
    - estilos-Inicio.css: Estilos de la página principal.
    - estilos-Categoria.css: Estilos específicos de la sección de categorías.
    - estilos-Distribuidoras.css: Estilos para la lista de distribuidores.
    - estilos-Datos de la distribuidora.css: Estilos para páginas individuales de cada distribuidora.

DISTRIBUIDORA

- Contiene páginas HTML de cada distribuidora:
    - Campero.html, Carpiteria.html, ICA.html, Incer.html
- Cada página tiene **información específica del negocio**, como descripción, productos, contacto e imágenes.

IMAGENES

- Carpeta principal para todos los recursos gráficos de la página:
    - logo_de_emprendehub.png: Logo de la marca.
- Subcarpetas por distribuidora:
    - **campero, carpinteria, ica, incer**: contienen imágenes de productos, logos y fotografías relevantes de cada distribuidora.
- Esta organización facilita **referencias rápidas desde HTML y CSS** y mantiene las imágenes ordenadas por categoría.

JAVASCRIPT

- Scripts organizados por funcionalidad y página:
    - Inicio.js: Funciones para la página principal, como sliders o animaciones.
    - Categoria.js: Manejo de interacciones dentro de la sección de categorías.
    - Distribuidor.js: Scripts comunes para manejar información dinámica de las distribuidoras.
    - Datos de la distribuidora.js: Funciones específicas para páginas individuales de distribuidoras.

WIREFRAMES

- Contiene **prototipos visuales de las páginas**, utilizados para planificar la disposición de los elementos antes de implementar el diseño final:
    - categoria.png, datos de la distribuidora.png, distribuidores.png, inicio.png

5.3 Organización y Flujo del Frontend

- El **HTML** define la estructura de la página y referencia los **archivos CSS y JS correspondientes**.
- El **CSS** controla la apariencia, estilos y animaciones de los elementos.
- El **JavaScript** gestiona la interacción del usuario, como carruseles, menús y validaciones.
- Las **imágenes** y recursos gráficos se cargan según la página y sección correspondiente, optimizando el rendimiento.
- Cada **distribuidora tiene su propia página HTML** y su CSS opcional, permitiendo mantener un estilo uniforme pero adaptado a cada negocio.

5.4 Ventajas de esta Arquitectura

1. **Modularidad:** Cada página y sección tiene archivos CSS y JS separados, facilitando el mantenimiento.
2. **Escalabilidad:** Es fácil agregar nuevas distribuidoras, categorías o secciones sin afectar el resto del proyecto.
3. **Claridad:** La estructura jerárquica de carpetas permite a nuevos desarrolladores entender rápidamente la organización.
4. **Optimización:** Separar imágenes por carpetas reduce errores y facilita la referencia en HTML y CSS.

6. Funcionamiento del Frontend

El frontend de la página web **EmprendeHub** está diseñado para ofrecer una **experiencia de usuario fluida, interactiva y visualmente atractiva**, permitiendo la navegación entre categorías, distribuidores y páginas individuales con animaciones y elementos interactivos que facilitan la exploración de contenido.

El **funcionamiento del frontend** se basa en la interacción entre tres componentes principales: **HTML, CSS y JavaScript**, además de la correcta organización de **imágenes y recursos gráficos**. A continuación, se describe detalladamente cómo funciona cada parte del frontend y cómo interactúan entre sí.

6.1 Flujo General del Frontend

1. **Carga inicial de la página**
    - Cuando el usuario accede a index.html, el navegador interpreta la estructura HTML y carga los estilos desde CSS/estilos-Inicio.css.
    - Las imágenes de la carpeta IMAGENES/ se cargan según la sección mostrada en la página (por ejemplo, el logo de la plataforma o banners principales).
    - Los scripts de JavaScript (Inicio.js) se ejecutan para activar animaciones iniciales, sliders o cualquier componente dinámico presente en la página principal.
2. **Navegación entre secciones**
    - Los usuarios pueden navegar a otras páginas como Categoria.html o Distribuidoras.html a través de enlaces del menú principal.
    - Cada página HTML referencia su **propio archivo CSS** para mantener estilos específicos y su **archivo JS correspondiente** para funcionalidades dinámicas.
    - Esto permite que cada sección sea independiente y modular, evitando conflictos entre estilos y scripts de distintas páginas.
3. **Interacción con los elementos de la página**
    - Botones, enlaces y tarjetas de productos utilizan **eventos JavaScript** para responder a la acción del usuario.
    - Ejemplos de interacciones:
        - Carruseles de productos o distribuidores (mover tarjetas al hacer clic en botones “Siguiente” o “Anterior”).
        - Formularios de contacto o búsqueda que validan campos antes de enviar la información.
        - Animaciones visuales al pasar el mouse sobre botones, tarjetas o enlaces.

6.2 Funciones JavaScript Principales

El frontend utiliza **archivos JavaScript específicos por sección**, permitiendo modularidad y facilidad de mantenimiento.

6.2.1 Inicio.js

- Funcionalidades:
    - Activación de banners animados en la página principal.

- Efectos de transición en botones y enlaces.
- Control de sliders o carruseles iniciales de productos destacados.

6.2.2 Categoria.js

- Funcionalidades:
  - Gestión del carrusel de categorías de productos.
  - Aplicación de filtros visuales por tipo de producto.

Ejemplo de carrusel de categorías:

```
const carrusel = document.querySelector('.carrusel');
const tarjetas = document.querySelectorAll('.tarjeta');
let index = 0;

function actualizarCarrusel() {
    const anchoTarjeta = tarjetas[0].offsetWidth + 20;
    carrusel.style.transform = `translateX(-${index * anchoTarjeta}px)`;
}

document.querySelector('#next').addEventListener("click", () => {
    if(index < tarjetas.length - 1) index++;
    actualizarCarrusel();
});

document.querySelector('#prev').addEventListener("click", () => {
    if(index > 0) index--;
    actualizarCarrusel();
});
```

6.2.3 Distribuidor.js

Este archivo gestiona la **visualización general de todas las distribuidoras** y su interacción con el usuario mediante un mapa y filtros dinámicos.

Principales funciones:

1. **Inicialización del mapa (initMap)**

- Se crea un objeto google.maps.Map centrado en coordenadas predeterminadas de la ciudad (lat: -16.529, lng: -68.169) con un nivel de zoom inicial de 12.
- Se agregan botones de zoom (+/-) que modifican el nivel de zoom del mapa dinámicamente.
- Cada distribuidora se representa con un **marcador** en el mapa, utilizando coordenadas almacenadas en los atributos data-coordenadas de cada tarjeta de distribuidora.

2. **Actualización de marcadores (actualizarMarcadores)**
   - Antes de actualizar, se eliminan los marcadores antiguos del mapa para evitar duplicaciones.
   - Se recorre cada tarjeta de distribuidora visible en la lista y se crea un marcador con título y ventana de información (InfoWindow) que muestra el nombre, dirección y categoría de la distribuidora.
   - Cada marcador tiene un evento de **click** que despliega la ventana de información asociada.
3. **Filtrado de distribuidoras (filtrarResultados)**
   - Permite filtrar las distribuidoras por:
     - **Nombre:** coincidencia parcial con el texto ingresado en el campo de búsqueda.
     - **Zona:** coincidencia con la zona seleccionada en un menú desplegable.
     - **Categoría:** coincidencia con la categoría seleccionada.
     - **WhatsApp:** opción para mostrar solo distribuidores que tengan contacto por WhatsApp.
   - Las tarjetas que cumplen los criterios se muestran, mientras que las que no cumplen se ocultan (display: none).
   - Se actualiza el **contador de resultados** y los **marcadores del mapa** para reflejar únicamente las distribuidoras visibles.
4. **Eventos y localStorage**
   - Los campos de búsqueda, los filtros y el botón de WhatsApp tienen eventos asociados que llaman a la función filtrarResultados.
   - Se almacenan criterios de búsqueda previos en localStorage para mantener consistencia cuando el usuario recarga la página.

Ejemplo de interacción de botones de zoom y filtrado:

```
document.getElementById("zoomMas").addEventListener("click", () => mapa.setZoom(mapa.getZoom() + 1));
document.getElementById("zoomMenos").addEventListener("click", () => mapa.setZoom(mapa.getZoom() - 1));
```

```
inputBusqueda.addEventListener("input", filtrarResultados);
selectZona.addEventListener("change", filtrarResultados);
selectCategoria.addEventListener("change", filtrarResultados);
```

6.2.4 Datos de la distribuidora.js

Este archivo se encarga de la **interactividad en las páginas individuales de cada distribuidora**, gestionando la ubicación en el mapa y la conexión con WhatsApp.

Principales funciones:

1. **Inicialización del mapa individual (initMapaReal)**
    - El mapa se centra en la ubicación específica de la distribuidora, obtenida de un atributo data-coordenadas en el contenedor del mapa.
    - Se coloca un **marcador único** indicando la ubicación exacta de la empresa.
    - Se configuran botones de zoom para acercar o alejar la vista del mapa de forma dinámica.
2. **Botón de contacto WhatsApp**
    - Se obtiene el número de teléfono desde un elemento con id telefono-principal.
    - Se normaliza el número eliminando caracteres no numéricos y asegurando que tenga el **código de país (591)** si corresponde a Bolivia.
    - Al hacer clic en el botón de WhatsApp, se abre la **URL oficial de WhatsApp** en una nueva pestaña, permitiendo al usuario iniciar una conversación directa con la distribuidora.
3. **Validaciones y seguridad**
    - Se verifican elementos del DOM (teléfono y botón) para evitar errores si algún elemento no existe en la página.
    - Se manejan errores y alertas para asegurar que la funcionalidad de contacto no falle.

Ejemplo de inicialización del mapa individual:

```
const ubicacion = { lat: latitud, lng: longitud };

// Crear mapa
map = new google.maps.Map(mapaDiv, {
    center: ubicacion,
    zoom: zoomActual,
});

// Crear marcador
new google.maps.Marker({
    position: ubicacion,
    map: map,
    title: "Ubicación de la empresa"
});
```

Ejemplo de configuración del botón WhatsApp:

```
boton.addEventListener("click", (e) => {
    e.preventDefault();
    const url = "https://wa.me/" + telefonoConPais;
    window.open(url, "_blank");
});
```

6.2.5 Integración y Flujo de Trabajo

- Distribuidor.js maneja **la vista general de todas las distribuidoras**, incluyendo mapa interactivo y filtrado.
- Datos de la distribuidora.js gestiona **páginas individuales**, donde el usuario ve información específica, ubicación exacta y puede contactar directamente vía WhatsApp.
- Ambos scripts están diseñados para ser **modulares y complementarios**, permitiendo una experiencia consistente y fluida desde la lista general hasta la información individual de cada distribuidora.

7. Uso de CSS y Diseño del Frontend

El **CSS (Cascading Style Sheets)** en la página web **EmprendeHub** se utiliza para controlar la **presentación visual**, la **disposición de los elementos**, las **tipografías**, los **colores corporativos** y las **animaciones** que mejoran la experiencia de usuario.

Cada página HTML tiene su **archivo CSS específico**, lo que permite **modularidad y mantenimiento sencillo**. Además, se usan **pseudo-clases, pseudo-elementos y transiciones CSS** para generar efectos interactivos y atractivos.

7.1 Organización de Archivos CSS

Archivo CSS	Página asociada	Función principal
estilos-Inicio.css	index.html	Define estilos de la página principal, banners, menú y botones.
estilos-Categoria.css	Categoria.html	Maneja la presentación de categorías, tarjetas de productos y carrusel.
estilos-Distribuidoras.css	Distribuidoras.html	Controla la lista de distribuidores, filtros y tarjetas de resultados.
estilos-Datos de la distribuidora.css	DISTRIBUIDORA/*.html	Estilos de páginas individuales, galerías de imágenes, mapa y botón WhatsApp.

7.2 Diseño General

1. **Tipografía y colores**
    - Se utiliza la fuente **Inter**, importada desde Google Fonts para todos los textos.
    - Colores corporativos:
        - Azul claro para encabezados y botones (#4dd0e1).
        - Blanco para fondos principales.
        - Gris para textos secundarios (#333333).

Ejemplo de estilos globales:

```css
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: 'Inter', sans-serif;
    background-color: #ffffff;
    color: #333333;
    line-height: 1.5;
}
```

2. **Cabecera y menú de navegación**
    - La cabecera (header) utiliza **flexbox** para alinear el logo y menú horizontalmente.
    - Los enlaces de navegación cambian de color al pasar el mouse, usando **pseudo-clases :hover**.

```css
header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 50px;
    background-color: #4dd0e1;
}

header nav a {
    text-decoration: none;
    color: white;
    margin-left: 20px;
    transition: color 0.3s ease;
}

header nav a:hover {
    color: #ffffff;
    text-shadow: 0px 0px 5px #00000050;
}
```

7.3 Estilos de Tarjetas y Contenedores

Las secciones de categorías y distribuidores utilizan **tarjetas (.tarjeta)** con sombras, bordes redondeados y animaciones al pasar el mouse.

```css
.tarjeta {
    background-color: #f9f9f9;
    border-radius: 15px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.1);
    overflow: hidden;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}

.tarjeta:hover {
    transform: scale(1.05);
    box-shadow: 0 8px 16px rgba(0,0,0,0.2);
}
```

7.4 Animaciones CSS

Se aplican **animaciones con @keyframes** y transiciones para mejorar la experiencia visual.

1. **Fade-in al cargar elementos:**

```css
@keyframes fadeIn {
    from { opacity: 0; }
    to { opacity: 1; }
}

.tarjeta {
    animation: fadeIn 0.8s ease-in-out;
}
```

2. **Transición suave de botones:**

```css
.boton {
    padding: 10px 20px;
    border: none;
    border-radius: 10px;
    background-color: #4dd0e1;
    color: white;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.3s ease;
}

.boton:hover {
    background-color: #36bfcf;
    transform: scale(1.05);
}
```

7.5 Diseño de Carrusel de Categorías

- El carrusel utiliza **flexbox** para alinear las tarjetas en fila horizontal.
- Se aplican **transiciones en transform** para el movimiento de izquierda a derecha al usar los botones de navegación.

```css
.carrusel {
    display: flex;
    overflow: hidden;
    gap: 20px;
}

.carrusel .tarjeta {
    min-width: 250px;
    flex-shrink: 0;
    transition: transform 0.3s ease;
}
```

7.6 Estilos de la Sección de Distribuidoras

1. **Filtros y barra de búsqueda:**

```css
.filtro-zona, .filtro-categoria {
    margin-bottom: 20px;
}

.filtro-zona select, .filtro-categoria select, .campo-busqueda input {
    padding: 8px 12px;
    border-radius: 8px;
    border: 1px solid #ccc;
    transition: border-color 0.3s ease;
}

.campo-busqueda input:focus {
    border-color: #4dd0e1;
    outline: none;
}
}
```

2. **Lista de resultados y tarjetas:**

```css
.lista-resultados {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 20px;
    margin-top: 20px;
}
```

7.7 Estilos para Páginas Individuales de Distribuidoras

1. **Galería de imágenes:**

```css
.galeria {
    display: flex;
    gap: 10px;
    overflow-x: auto;
}

.galeria img {
    border-radius: 10px;
    transition: transform 0.3s ease, box-shadow 0.3s ease;
}
```

2. **Mapa de ubicación:**

```css
#mapaReal {
    width: 100%;
    height: 400px;
    border-radius: 15px;
    margin-top: 20px;
}
```

3. **Botón de WhatsApp:**

```css
.boton-whatsapp {
    background-color: #25d366;
    color: white;
    padding: 12px 20px;
    border-radius: 10px;
    cursor: pointer;
    text-align: center;
    font-weight: bold;
    transition: background-color 0.3s ease, transform 0.3s ease;
}

.boton-whatsapp:hover {
    background-color: #1ebe5d;
    transform: scale(1.05);
}
```

7.8 Ventajas de la Implementación de CSS

1. **Modularidad:** Cada página tiene su archivo CSS, lo que facilita mantenimiento y escalabilidad.
2. **Interactividad visual:** Animaciones, transiciones y efectos hover mejoran la experiencia del usuario.
3. **Diseño responsivo:** Flexbox y Grid permiten que las páginas se adapten a diferentes tamaños de pantalla.
4. **Consistencia visual:** Colores, tipografías y estilos uniformes en toda la plataforma.
5. **Optimización de rendimiento:** Carga de estilos específica por página, evitando conflictos y sobrecarga de CSS innecesario.

8. Integración y Uso de la API de Google Maps

La página web **EmprendeHub** integra la **API de Google Maps** para mostrar la ubicación de las distribuidoras y mejorar la experiencia del usuario en la búsqueda de negocios. Esta sección explica cómo se implementa y funciona la API dentro del frontend.

8.1 Funcionalidad de la API

1. **Mapa general de distribuidores**
    - En la página Distribuidoras.html, se utiliza Google Maps para mostrar un **mapa centralizado** de la ciudad.
    - Cada distribuidora se representa mediante un **marcador**, que contiene información de nombre, dirección y categoría.
    - Los marcadores se actualizan dinámicamente según los filtros de búsqueda (nombre, zona, categoría y WhatsApp).
2. **Mapa individual de distribuidora**
    - Cada página de distribuidora (DISTRIBUIDORA/*.html) tiene un mapa centrado en la **ubicación específica del negocio**.
    - Se coloca un **marcador único** indicando la posición exacta de la empresa.
    - Los botones de zoom permiten al usuario acercar o alejar la vista del mapa según sus necesidades.

8.2 Integración en el Frontend

1. **Incluir la API en HTML**
    - Se incluye el script de Google Maps en el <head> o al final del <body> de las páginas correspondientes:

- ```<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAi7iFNC3cGHMxbw1FUi0mMuEs80vzmxw4&callback=initMap" async defer></script>```

  TU_API_KEY debe reemplazarse por la **clave de API de Google Maps**.
- callback=initMap asegura que se ejecute la función de inicialización cuando la API haya cargado completamente

2. **Inicialización del mapa general (Distribuidor.js)**

```
let mapa; // variable global
let marcadores = [];

function initMap() {
    const mapaDiv = document.getElementById("mapaReal");

    mapa = new google.maps.Map(mapaDiv, {
        center: { lat: -16.529, lng: -68.169 },
        zoom: 12
    });

    document.getElementById("zoomMas").addEventListener("click", () => mapa.setZoom(mapa.getZoom() + 1));
    document.getElementById("zoomMenos").addEventListener("click", () => mapa.setZoom(mapa.getZoom() - 1));

    actualizarMarcadores();
}
```

- mapa es la variable global que almacena el objeto Google Maps.
- marcadores contiene todos los marcadores visibles en el mapa.
- Los botones de zoom permiten interacción dinámica sin recargar la página.

3. **Actualización dinámica de marcadores según filtros**

```
function actualizarMarcadores() {
    marcadores.forEach(m => m.setMap(null));
    marcadores = [];

    const tarjetas = document.querySelectorAll(".tarjeta-distribuidora");
    tarjetas.forEach(tarjeta => {
        if(tarjeta.style.display === "none") return;
        const coordenadas = tarjeta.getAttribute("data-coordenadas");
        if(!coordenadas) return;
        const [lat, lng] = coordenadas.split(",").map(Number);

        const marcador = new google.maps.Marker({
            position: { lat, lng },
            map: mapa,
            title: tarjeta.querySelector("h3").textContent
        });

        const info = new google.maps.InfoWindow({
            content: `<strong>${tarjeta.querySelector("h3").textContent}</strong><br>
                      ${tarjeta.querySelector("p:nth-of-type(1)").textContent}<br>
                      ${tarjeta.querySelector("p:nth-of-type(2)").textContent}`
        });

        marcador.addListener("click", () => info.open(mapa, marcador));
        marcadores.push(marcador);
    });
}
```

- Los marcadores se muestran únicamente para **tarjetas visibles**, asegurando que el mapa refleje los filtros aplicados por el usuario.

8.3 Mapa Individual de Distribuidora (Datos de la distribuidora.js)

```
let map;
let zoomActual = 15;

function initMapaReal() {
    const mapaDiv = document.getElementById("mapaReal");
    const coords = mapaDiv.getAttribute("data-coordenadas").split(",");
    const ubicacion = { lat: parseFloat(coords[0]), lng: parseFloat(coords[1]) };

    map = new google.maps.Map(mapaDiv, {
        center: ubicacion,
        zoom: zoomActual
    });

    new google.maps.Marker({
        position: ubicacion,
        map: map,
        title: "Ubicación de la empresa"
    });

    document.getElementById("zoomMas").addEventListener("click", () => {
        zoomActual++;
        map.setZoom(zoomActual);
    });

    document.getElementById("zoomMenos").addEventListener("click", () => {
        zoomActual--;
        map.setZoom(zoomActual);
    });
}

window.onload = initMapaReal;
```

- Este código asegura que cada distribuidora tenga un **mapa personalizado**, centrado en su ubicación, con marcador y control de zoom.

8.4 Ventajas del Uso de Google Maps

1. **Interactividad:** Los usuarios pueden explorar la ubicación de los distribuidores de forma visual y dinámica.
2. **Actualización dinámica:** Los marcadores reflejan los filtros en tiempo real, evitando recargas de página.
3. **Integración sencilla:** La API se conecta directamente con atributos HTML (data-coordenadas) y scripts JS.
4. **Mejora la experiencia de usuario:** Facilita encontrar distribuidoras por zona y categoría y permite interacción con mapas de forma intuitiva.
5. **Escalabilidad:** Se pueden agregar nuevas distribuidoras con coordenadas sin modificar el código principal, simplemente agregando nuevas tarjetas en HTML.