



## Portal Interactivo

con Reproductor de música

Blog

Noticias

Agustín Benítez López

Marcos Jimenez García

Pedro Utrera Marín

Daniel Gómez Ortiz



# ÍNDICE

- **Introducción**   
- **Landing Page**
  -  [Landing Page HTML](#)
  -  [Landing Page CSS](#)
- **Main Page Reproductor**
  -  [Main Page HTML](#)
  -  [Main Page CSS](#)
  -  [Lista Canciones JS](#)
  -  [reproductor JS](#)
  -  [data JSON Canciones](#)
  -  [Themes CSS](#)
  -  [Settings JS](#)
  -  [theme-loader JS](#)
  -  [Tus Favoritos](#)
  -  [Login | Register](#)
- **Blog**
  -  [Blog HTML](#)
  -  [Blog CSS](#)
  -  [Blog JS](#)
  -  [Blog Json](#)
- **News**
  -  [News HTML](#)
  -  [News CSS](#)
  -  [News JS](#)
- **Otras Paginaciones**
  -  [Uso del CSS](#)
  -  [Uso del HTML](#)
- **Bibliografía**

# Introducción

Pensamos la idea sobre la que crear el proyecto,  
Decidimos crear un Portal Interactivo Musical

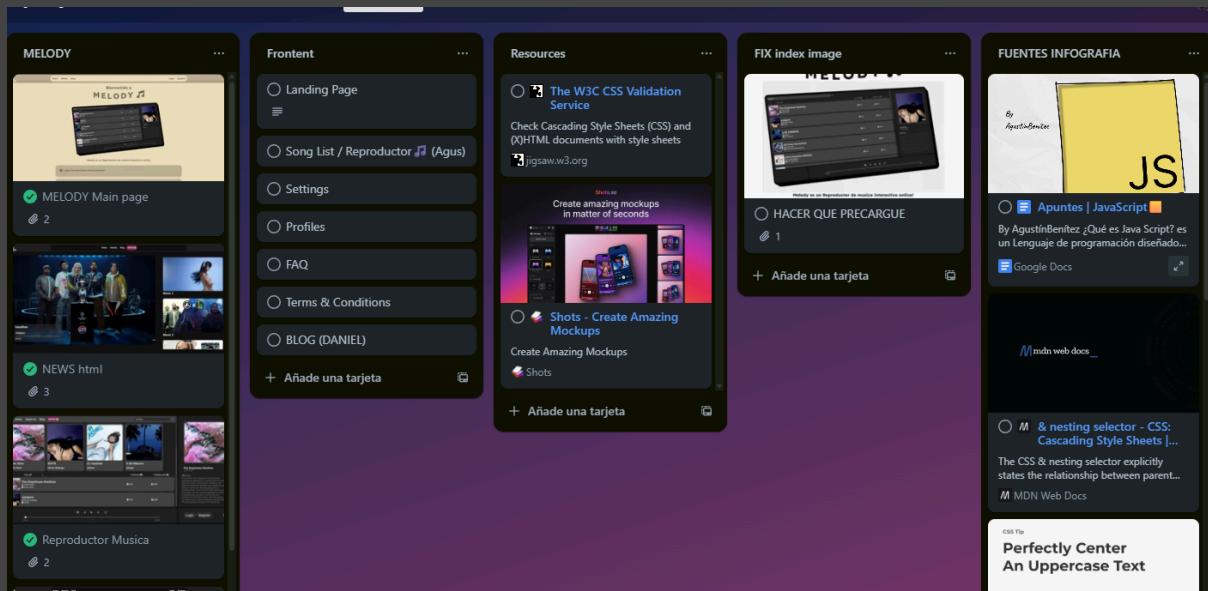
Usamos el nombre Melody que viene de melodía  
y diseñamos el LOGOTIPO con una M en forma de corchea ♩

Lo diseñamos en **figma**  y lo podemos así exportar a SVG, formato Vectorial

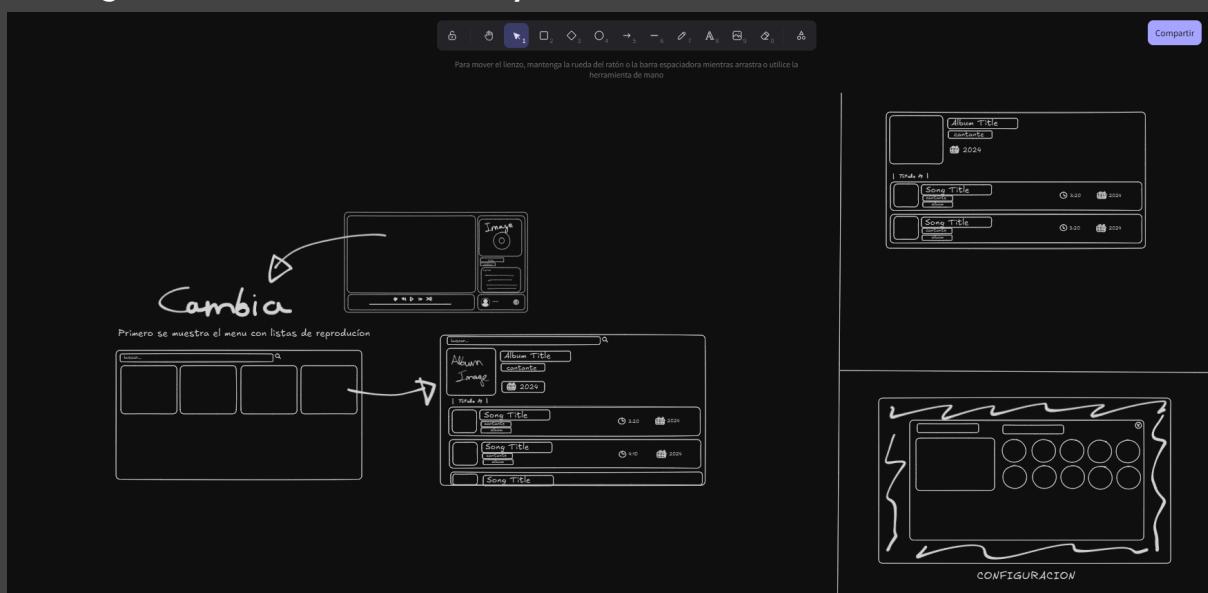


Queremos crear una interfaz minimalista  
para ello usaremos border radius 15px  
y una gama de colores escala de grises pero también crearemos varios temas  
con distintos colores que podrás cambiar en los ajustes a gusto del usuario.

Usamos **trello** para organizar el proyecto



también usamos **Escalidraw** y **figma** para diseñar la estructura y los diagramas de funcionalidades y como funcionara todo.





# Landing Page

La landing Page se encargará **Marcos** creando el **Index.html**

Primero creo la estructura con el **HTML 5** con un nav / main / footer

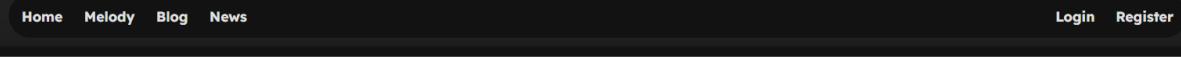
```
● ● ●
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Melody | Home</title>
7  </head>
8  <body>
9      <a href="/index.html">
10         </a>
11     <header id="header-main">
12         <nav>
13             </nav>
14     </header>
15
16     <main>
17         <h1>Bienvenido a</h1>
18     </main>
19     <footer>
20         <nav>
21             <div id="settings" class="quit-mobile">
22
23             </div>
24             <div class="privacy-and-about">
25                 <a href="/pages/contribuidores.html">Contribuidores</a>
26                 <a href="/pages/contactar.html">Contactanos</a>
27                 <a href="/pages/terms-&-conditions.html">Terminos Y Condiciones</a>
28             </div>
29
30         </nav>
31     </footer>
32 </body>
33 </html>
```

Luego creó el nav y le doy IDs y clases para luego poder utilizar el mismo Style en todas las páginas menos en el reproductor

```
● ● ●
1  <header id="header-main">
2      <nav>
3          <div id="home-melody">
4              <a href="/index.html">Home</a>
5              <a href="/pages/main-page.html">Melody</a>
6              <a href="/pages/blog.html">Blog</a>
7              <a href="/pages/news.html">News</a>
8          </div>
9          <div class="login-register quit-mobile">
10             <a href="">Login</a>
11             <a href="">Register</a>
12         </div>
13
14     </nav>
15 </header>
```

Luego creó el Footer y le asignó IDs para poder identificarlo para luego aplicarle estilos y poder reciclar entre el nav y el footer.

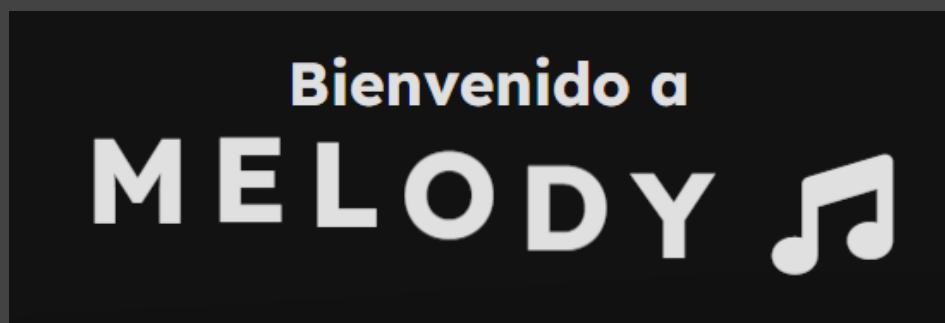
Ahora creo el **Css** de del Nav junto al Footer, para poder reutilizar los estilos de los botones.



Para que los menús sean Responsable modifique los botones para que puedan adaptarse según la resolución de la pantalla.

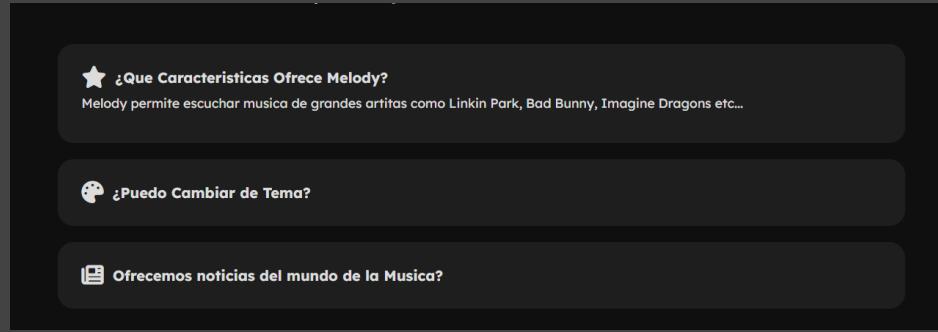
Luego empiezo con la estructura inicial de la Landing page, en la landing page aparecerá un título que ponga “Melody” con un estilo particular, aparte tendrá un icono al final.

Para el movimiento particular usaré una virtud del CSS 3 que me permite ir ciclando de `<span>` en `<span>` para aplicar un movimiento, en este caso será un movimiento de arriba a abajo.



Luego crearemos la main Image, la gracia principal es enseñar la UI y con un simple vistazo, para ello usar la Web [Shots.so](#), con ella es muy fácil crear mockups (imágenes para exponer de forma sencilla y elegante una web o app).

Otro añadido serían los acordeones de explicaciones. Un acordeon es una etiqueta especial de HTML 5. Esta etiqueta es capaz de automáticamente crear un desplegable sin usar JS.



**Lo siguiente será añadir un mini expositor de las UI en distintos colores, para exponer los colores más destacables pero esto lo hago cuando mi compañero termina de hacer este apartado de themes en main-page.html.**

**Otros apartados en la Landing Page son los colores disponibles:**



**Estos colores se podrán ampliar más adelante.**

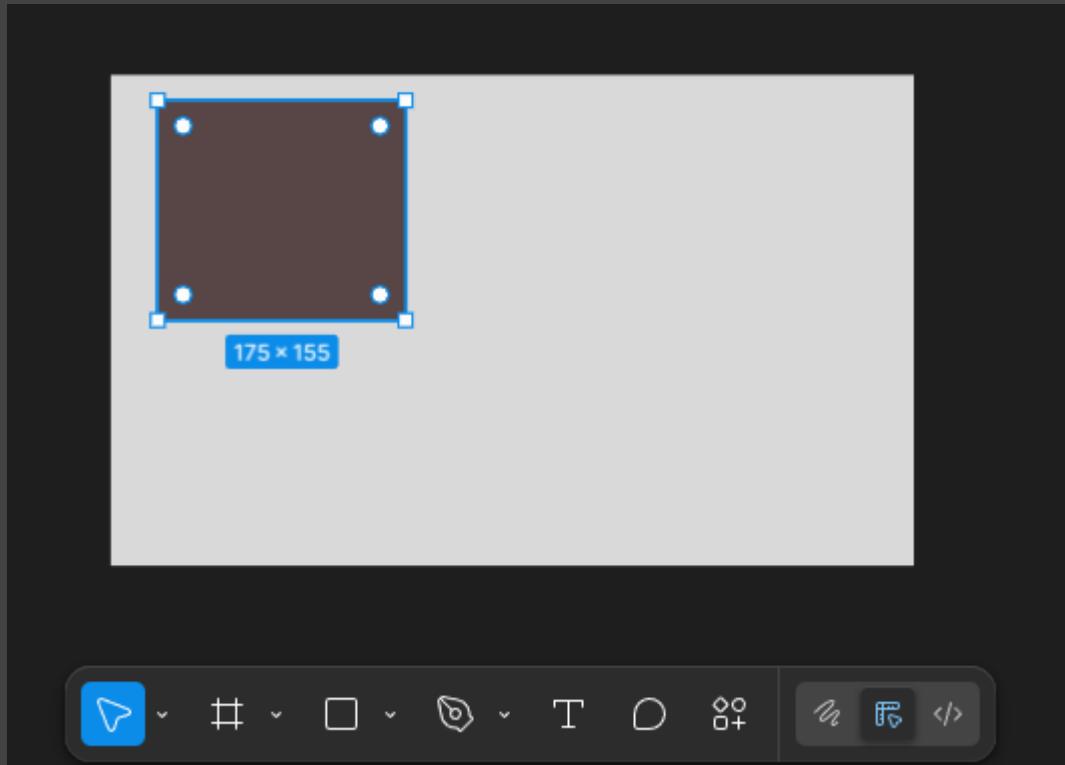
**Por último, enseñamos la últimas noticias de la sección News:**



# Main Page | Reproductor Música

De este apartado se Encargará Agustín  el archivo se llamará **main-page.html**  
Esta página se trata de un Reproductor de música en el que podrás elegir la música que escuchas , cambiar el tema/Color de todo el Sitio Web en los ajustes.

Primero Diseño en **figma**  como quiero que se vea la interfaz

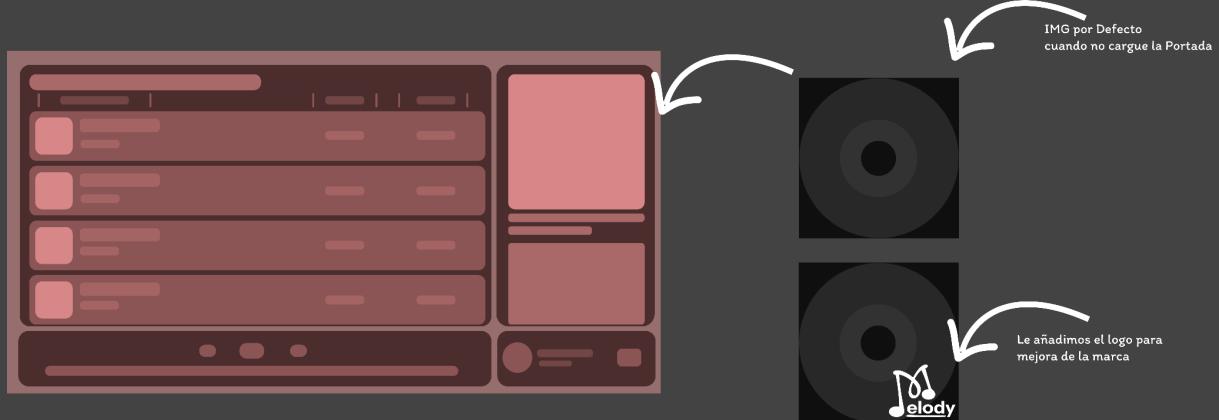


serán 4 sections en un main y lo estructuramos con GRID 



esta será la estructura un section con info canción otro canciones otro controles y otro de settings

quiero meter una lista(no lista literal) con diferentes divs de canciones que tendrán la imagen de Portada, título, cantante, álbum, duración y fecha add



en el section de la derecha aparecerá la información de la canción que selecciones como:  
portada, título, artista y Letra de canción

en el section de abajo izquierda los controles de reproducción  
pause, volumen, barra de reproducción con el tiempo → 00:00 ----- 3:00

y abajo derecha botón de ajustes para cambiar el tema y iniciar sesión

Luego creo la estructura en el html **main-page.html**

```
● ● ●
1  <!DOCTYPE html>
2  <html lang="es">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Reproductor Melody</title>
7  </head>
8  <body>
9      <main>
10         <!-- Lista de canciones -->
11         <section id="song-list" class="item">
12             <header id="header-main-page">
13                 <nav>
14                 </nav>
15                 <div id="search">
16                     <input type="text" placeholder="buscar..."><i class="fa-solid fa-magnifying-glass"></i>
17                 </div>
18             </header>
19             <section id="search-main-page">
20                 <article id="albums-list">
21                     <!-- ALBUMS LIST -->
22                 </article>
23                 <article id="list-container">
24
25                     </article>
26                 </section>
27             </section>
28             <!-- Informacion de la cancion -->
29             <section id="song-info" class="item">
30             </section>
31
32             <!-- Apartado de cuenta y ajustes -->
33             <section id="account-info" class="item">
34             </section>
35
36             <!-- Panel de control de la cancion -->
37             <section id="control-panel" class="item">
38
39             </section>
40         </main>
41     </body>
42 </html>
```

Luego creo el CSS **main.css** y estructuro el GRID  primero con \* {} reseteo el margen y padding de todos los divs etc...

y luego al main estructuro el Grid con las columnas y filas

```
● ● ●
1 /* ----- STYLES GENERALES -----*/
2 *
3   margin: 0;
4   padding: 0;
5   box-sizing: border-box;
6 }
7
8 body {
9   background-color: var(--background-color);
10  font-family: var(--fuente-text);
11  color: var(--font-color);
12 }
13
14 main {
15   display: grid;
16   grid-template-columns: 1fr 1fr 1fr 1fr 0.2fr 1fr;
17   grid-template-rows: 1fr 1fr 1fr 0.5fr;
18   gap: 10px;
19   margin: 10px;
20   width: 98.5vw;
21   height: 97dvh;
22 }
```

```
● ● ●
1 /* -----SECTIONS PRINCIPALES DEL GRID class.items----- */
2 .item:nth-child(1) {
3   grid-area: 1 / 1 / 2 / 2;
4   grid-column: 1 / 5;
5   grid-row: 1 / 4;
6 }
7
8 .item:nth-child(2) {
9   grid-area: 1 / 5 / 2 / 6;
10  grid-column: 5 / 8;
11  grid-row: 1 / 4;
12 }
13
14 .item:nth-child(3) {
15   grid-area: 4 / 2 / 5 / 3;
16   grid-column: 5 / 8;
17 }
18
19 .item:nth-child(4) {
20   grid-area: 4 / 1 / 5 / 2;
21   grid-column: 1 / 5;
22 }
```

he creado un fichero CSS llamado **themes.css** donde creo variables de color para poder usar los mismos colores en todas las páginas y que sea más fácil el cambio de colores:

```
1 :root {
2   --primary-color: #000;
3   --secondary-color: #fff;
4   --background-color: #121212;
5   --section-color: #212121;
6   --button-color: #303030;
7   --item-list: #424242;
8   --buscador: #424242;
9   --font-color: #e0e0e0;
10  --font-secondary-color: #818181;
11  --font-tertiary-color: #424242;
12  --blur-transparent: #4d4d4d53;
13  --salir-cancelar: #892557;
14  --salir-cancelar-hover: #611b3e;
15  /* -----fuente-----*/
16  --fuente-text: "Lexend", Arial, Helvetica, sans-serif;
17 }
```

y también meto la fuente en una variable para poder asignarlo rápido y que sea más fácil cambiarlo como si fuese una **CONSTANTES** de un lenguaje de programación .

y estas variables las usare para cada vez que quiera poner un color como el background color pues uso esto.

Pero para ello en cada CSS que lo usemos lo vamos a Importar con un @Import



```
1  /* ----- THEMES -----*/
2  /* ----- Asignacion Variables de colores -----*/
3  @import url('/styles/themes.css');
```

Esto lo usarán también mis compañeros.

y usaremos flex



para posicionar dentro de los grids las cosas como Nav, canciones etc...  
importare el css en el **HTML**



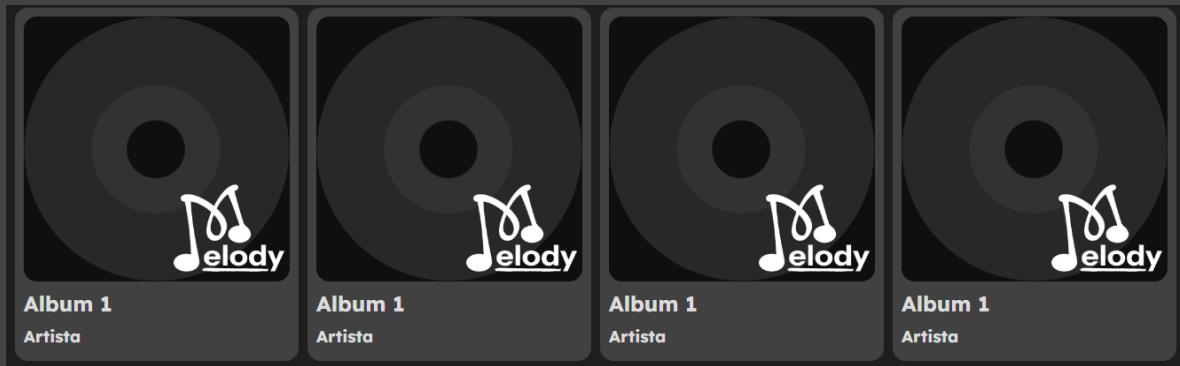
```
1  <link rel="stylesheet" href="/styles/main.css">
```

Creo los Divs de los Álbumes que luego reemplazamos el contenido como  
Portada , título, artista con el **JS** **JS** y creare un EventListener para que abra el  
Álbum que le hagas Click y muestre las canciones de ese Álbum



```
1  <article id="albums-list">
2      <!-- ALBUMS LIST -->
3      <div id="album-1" class="album">
4          
5          <h2>Album 1</h2>
6          <h3>Artista</h3>
7      </div>
8      <div id="album-2" class="album">
9          
10         <h2>Album 1</h2>
11         <h3>Artista</h3>
12     </div>
13     <div id="album-3" class="album">
14         
15         <h2>Album 1</h2>
16         <h3>Artista</h3>
17     </div>
18     <div id="album-4" class="album">
19         
20         <h2>Album 1</h2>
21         <h3>Artista</h3>
22     </div>
23 </article>
```

Y le añadimos styles con **CSS** y quedaria algo asi



también creo el Html para mostrar la lista de canciones

```
<article id="list-container">
  <nav id="nav-songs">
    | <span>Songs <i class="fa-solid fa-music"></i> </span> | <span id="separador-nav"></span> <span id="span-duration">| duration <i class="fa-solid fa-clock"></i> | </span>
  </nav>
  <div id="song-list-content">
    |<!-- Aquí van las canciones -->
  </div>
</article>
```

el nav de songs es este diseño pero creado con span para poder separarlo:

```
| Titulo ↗ | | duration (⌚) || Date Release 🎵 |
```

```
| <span>Songs <i class="fa-solid fa-music"></i> </span> | <span id="separador-nav"></span> <span id="span-duration">| duration <i class="fa-solid fa-clock"></i> | </span>
```

este es el Diseño es como quiero que quede:



en #song-list-content meteremos con JS las canciones 🎵



por ahora dejaré el container donde se añadirán

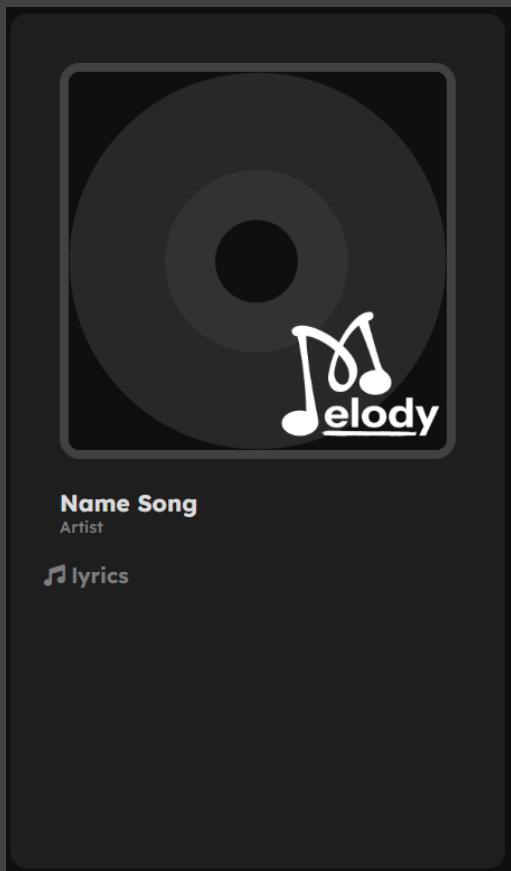
Ahora en el Section de #song-info meteré la estructura para que se vea así:



El html sin css todavía:

```
● ● ●
1  <!-- Informacion de la cancion -->
2  <section id="song-info" class="item">
3      <div class="song-info-container">
4          
5          <div id="song-info-text">
6              <h2 class="song-info-title">Name Song</h2>
7              <p class="song-info-artist">Artist</p>
8          </div>
9          <section id="lyrics-container">
10             <h3><i class="fa-solid fa-music"></i> lyrics</h3>
11             <p id="lyrics">
12                 <!-- Aqui es donde estara la letra de cada cancion-->
13             </p>
14         </section>
15     </div>
16 </section>
```

Y luego con el css se veria asi:

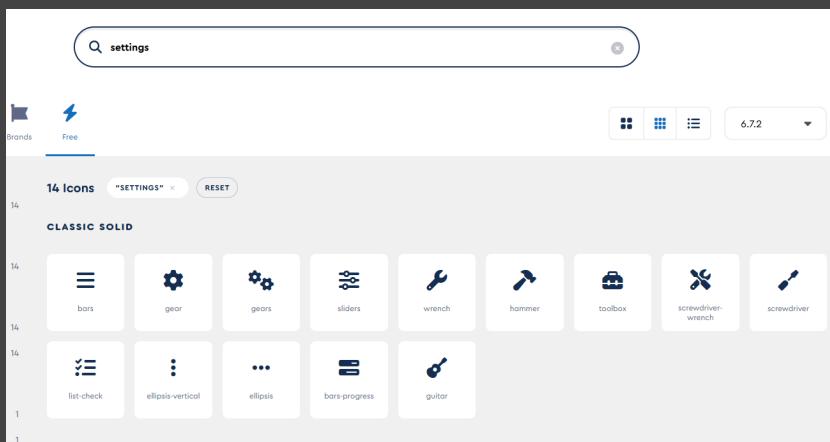


Usamos los Iconos de la librería [Awesome Fonts](#) que importamos en el html



```
1 <link href='https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.5.0/css/all.min.css' rel='stylesheet'/>
```

y es muy facil de usar buscamos en [AwesomeFonts.com](#) el icono que quiero usar y copiamos el code y lo añadimos donde queremos que esté



el code copiado es con una etiqueta creada por ellos llamada `<i>` y el class define el icono que es, estos iconos lo usamos en todas las páginas

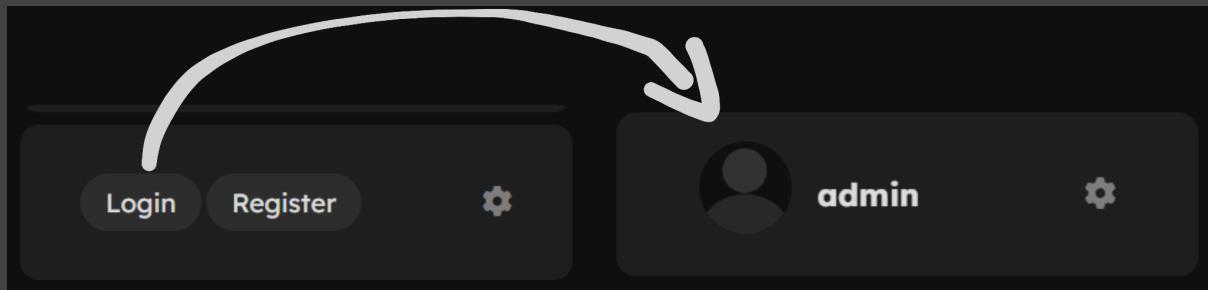
## Ahora creo los controles

```
● ● ●  
1 <!-- Panel de control de la canción -->  
2 <section id="control-panel" class="item">  
3   <div id="controls">  
4     <!-- like-->  
5     <button id="btn-fav">  
6       <i class="fa-solid fa-heart"></i>  
7     </button>  
8     <!-- play or stop-->  
9     <button id="btn-play">  
10       <i id="play-pause" class="fa-solid fa-play"></i>  
11     </button>  
12     <!-- volumen-->  
13     <button id="btn-volume">  
14       <i id="volume" class="fa-solid fa-volume-high"></i>  
15       <input type="range" id="volume-control-input" min="0" max="100" value="100">  
16     </button>  
17   </div>  
18   <div id="progress-bar">  
19     <input type="range" min="0" max="100" value="0" step="1" id="progress-bar-input">  
20     <div id="progress-bar-time">  
21       <span id="current-time">00:00</span>  
22       <span id="total-time">00:00</span>  
23     </div>  
24   </div>  
25 </section>
```

Todo esto después le añadiré funcionalidad con JS JS  
también creo el section de settings  
estará el div #account-login y el #account-info-text  
que cuando inicie sesión el usuario se ocultara el de #account-login  
con los botones y se mostrará el #account-info-text y se le reemplaza el  
contenido como img usuario y name <h2> con el JS JS

```
● ● ●  
1 <!-- Apartado de cuenta y ajustes -->  
2 <section id="account-info" class="item">  
3   <div id="account-login">  
4     <button id="btn-login" class="btn-login-register">Login</button>  
5     <button id="btn-register" class="btn-login-register">Register</button>  
6   </div>  
7  
8   <div id="account-info-text" class="hidden">  
9       
10    <h2>Account</h2>  
11  </div>  
12  <button id="btn-settings"><i class="fa-solid fa-gear"></i></button>  
13 </section>
```

esto quedara así con los styles CSS:



y creo el header con el nav

```
1 <header id="header-main-page">
2   <nav>
3     <button id="btn-settings" class="btn-nav-settings" onclick="ocultarModal()"><i class="fa-solid fa-gear"></i></button>
4     
5     <a href="/index.html">Home</a>
6     <a href="/pages/blog.html">Blog</a>
7     <a href="/pages/news.html">News</a>
8   </nav>
9   <div id="search">
10    <input type="text" placeholder="buscar... "><i class="fa-solid fa-magnifying-glass"></i>
11  </div>
12 </header>
```

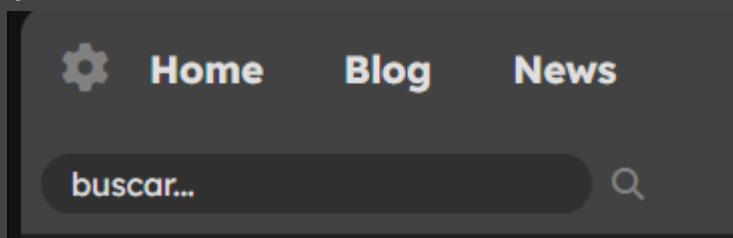
y con los styles css ->



el botón settings del header se oculta en el mediaquery normal y en móvil se mostrará este->

```
<button id="btn-settings" class="btn-nav-settings" onclick="ocultarModal()"><i class="fa-solid fa-gear"></i></button>
```

Creo un css con mediaquery para meter ahí todo lo del main.css  
y hago esto que se oculte en normal y en móvil se vea y se oculte el logo  
quedaria asi:



Ahora voy a empezar con todo el JavaScript JS



# JavaScript JS de main-page.html

creo un archivo js llamado **song-list-generator.js**  
y primero creo el apartado para reemplazar la información de los Álbumes  
Creo una **CONSTANTEs** con una **MATRIZ** de la información de todos los Álbumes  
porque no voy ha poner mas

```
● ● ●  
1 //----- CONSTANTES ALBUMS-----//  
2 const ALBUMS_LIST = [[{"Name": "From Zero", "URL": "/src/img/thumbnails/from-zero.webp", "Artist": "Linkin Park"},  
3 {"Name": "Debi Tirar Mas Fotos", "URL": "/src/img/thumbnails/baile_inolvidable_y_dbtf.webp", "Artist": "Bad Bunny"},  
4 {"Name": "11 razones", "URL": "/src/img/thumbnails/11Razones.webp", "Artist": "Aitana"},  
5 {"Name": "6 de febrero", "URL": "/src/img/thumbnails/6-de-febrero.webp", "Artist": "Aitana"}];  
6
```

la estructura del array es [Nombre, URL-Portada, Artista]  
luego capturo los divs de los Álbumes



```
1 //----- PAGINA DE INICIO REPRODUCTOR -----//  
2 //--- obtengo los elementos del html  
3 const album1 = document.getElementById("album-1");  
4 const album2 = document.getElementById("album-2");  
5 const album3 = document.getElementById("album-3");  
6 const album4 = document.getElementById("album-4");  
7 const albums = [album1, album2, album3, album4];
```

y luego creo una función donde recorro todos los Álbumes y le añado la info de  
la constante **ALBUMS\_LIST** = [---]  
Con esta función **loadAlbums()** lo llamo cuando cargue el DOM

```
● ● ●  
1 //----- FUNCIONES ALBUMS -----//  
2 function loadAlbums() {  
3     for (let i = 0; i < albums.length; i++) {  
4         let album = albums[i];  
5         const title = album.querySelector("h2");  
6         const img = album.querySelector("img");  
7         const author = album.querySelector("h3");  
8  
9         title.textContent = ALBUMS_LIST[i][0];  
10        img.src = ALBUMS_LIST[i][1];  
11        img.alt = ALBUMS_LIST[i][0];  
12        author.textContent = ALBUMS_LIST[i][2];  
13    }  
14 }  
15  
16 // Cargar los álbunes cuando el DOM esté listo  
17 document.addEventListener("DOMContentLoaded", loadAlbums);  
18
```

mediante el **for** va capturando de el  
álbum cada cosa que voy a  
reemplazar como el **h2** , **img** , **h3**  
y lo cambio con el índice del **FOR**

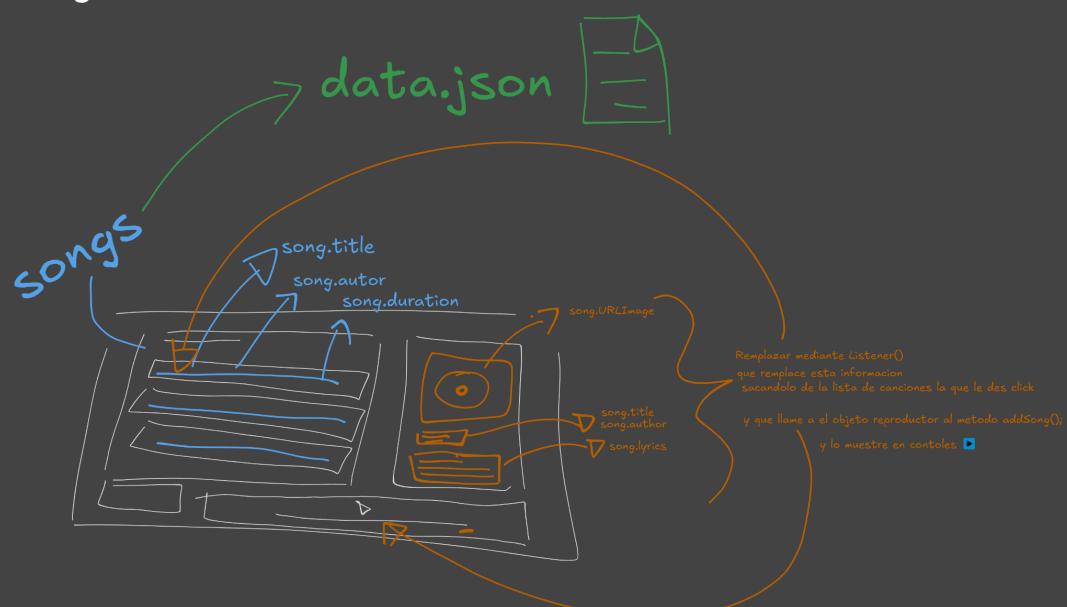
## Ahora quiero añadir la Funcionalidad de Cargar las CANCIONES

Para almacenar toda la información de las canciones usare **JSON{}** que luego con el **JS** lo iré iterando para crear objetos con los atributos de las canciones.

El **JSON{}** tendrá esta estructura :

```
{  
  "songs": [  
    {  
      "title": "The Emptiness Machine",  
      "artist": "...",  
      "album": "From Zero",  
      "genre": "Rock",  
      "releaseYear": 2024,  
      "duration": "3:20",  
      "audioUrl": "../src/songs/The-Emptiness-Machine.mp3",  
      "albumImageUrl": "/src/img/thumbnails/from-zero.webp",  
      "lyrics": "Your blades are sharpened with precision (precision)\\  
    },  
    {  
      "title": "vampire",  
      "artist": "Olivia Rodrigo",  
      "album": "GUTS",  
      "genre": "Pop",  
      "releaseYear": 2023,  
      "duration": "3:39",  
      "audioUrl": "../src/songs/vampire.mp3",  
      "albumImageUrl": "/src/img/thumbnails/GUTS.webp",  
      "lyrics": "Hate to give the satisfaction asking how you're doin  
    }  
  ]  
}
```

el diagrama de como funcionara es así:



el JS para gestionar esto es el siguiente:

Usare Fetch( la ruta del JSON)

Después, con .then(response => response.json()), convierte la respuesta en un objeto JavaScript para poder trabajar con ella.

Luego, con otro .then(data => {...}), accede a los datos ya convertidos (en este caso, una lista de canciones), y por cada canción:

```
//----- LEECTURA DE JSON Y MANEJO DE EVENTOS -----//
fetch("/src/data/data.json")
  .then((response) => response.json())
  .then((data) => {
    const songs = data.songs;
    const songList = document.getElementById("song-list-content");

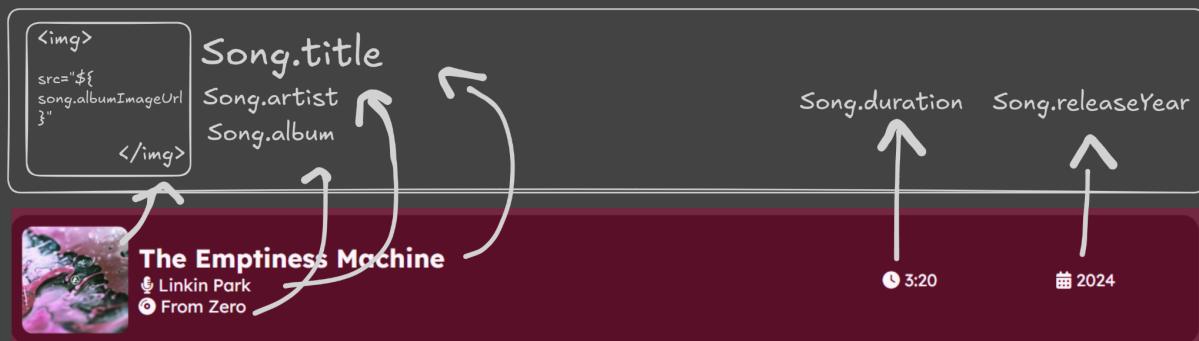
    songs.forEach((song) => {
      const songElement = document.createElement("div");
      cargarCanciones(song, songElement);
      // agrega el elemento a la lista de canciones
      songList.appendChild(songElement);
    });
  });
});
```

Luego creo un nuevo elemento HTML.

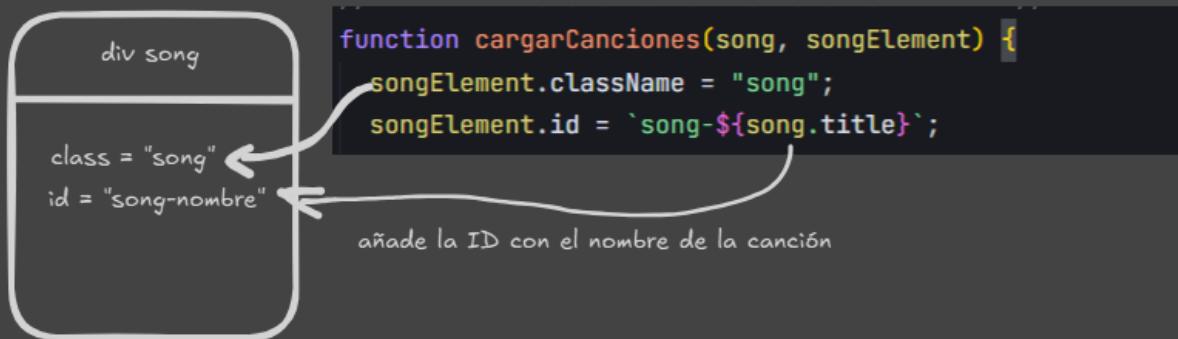
Llama a una función que se encarga de llenar ese elemento con los datos de la canción.

Finalmente, agrega ese elemento al HTML dentro de un contenedor con cierto id.

Una vez cargado el json llamaré a la función que creara este DIV(canción)



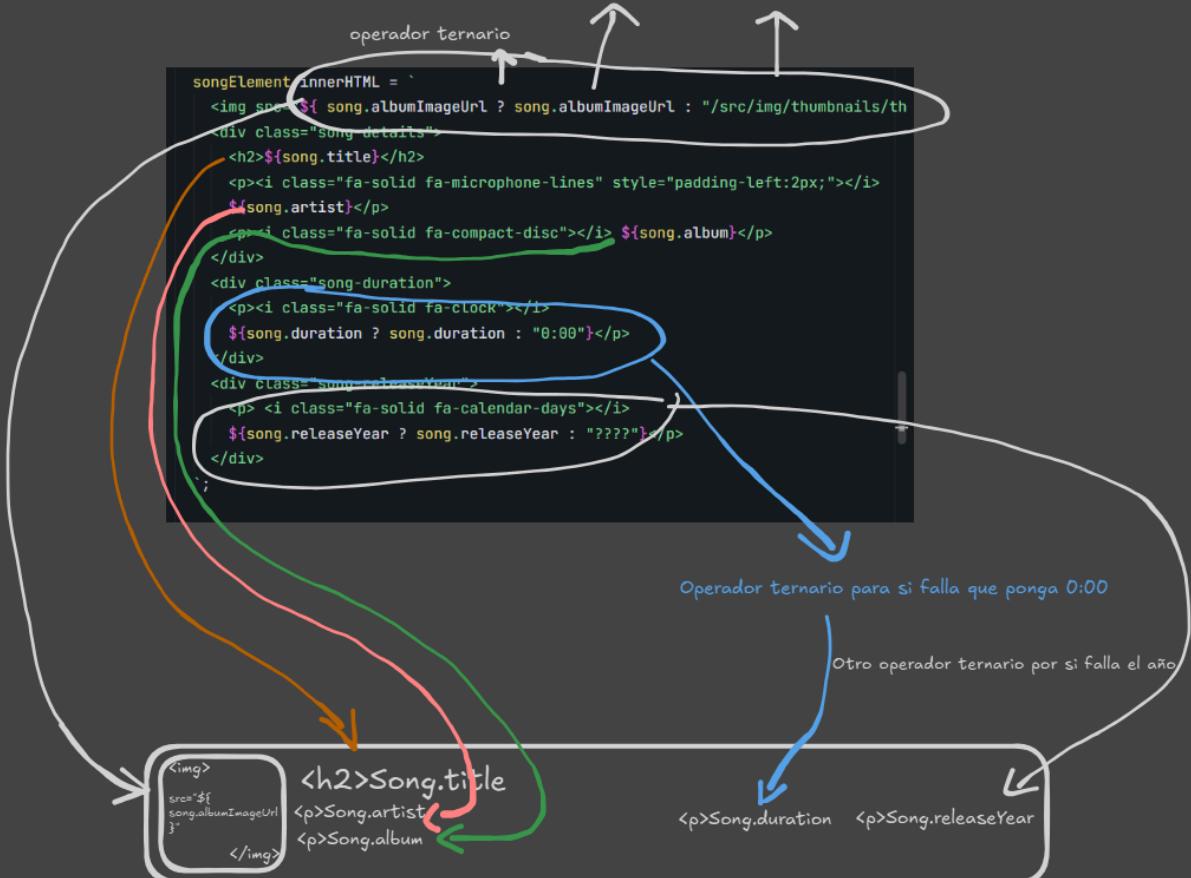
añade la class song a todas las canciones para que se vean igual



Primero añado la class song y ID generada con el prefijo song- y el nombre

y luego hago un innerHTML con todo el contenido  
explicación de cada apartado del code:

ruta del Json : ruta default por si falla



el code:

```
//----- FUNCION CARGAR CANCIONES -----//
function cargarCanciones(song, songElement) {
    songElement.className = "song";
    songElement.id = `song-${song.title}`;
    songElement.innerHTML = `
        
            <h2>${song.title}</h2>
            <p><i class="fa-solid fa-microphone-lines" style="padding-left:2px;"></i>
                ${song.artist}</p>
            <p><i class="fa-solid fa-compact-disc"></i> ${song.album}</p>
        </div>
        <div class="song-duration">
            <p><i class="fa-solid fa-clock"></i>
                ${song.duration ? song.duration : "0:00"}</p>
        </div>
        <div class="song-releaseYear">
            <p> <i class="fa-solid fa-calendar-days"></i>
                ${song.releaseYear ? song.releaseYear : "?????"}</p>
        </div>
    `;
}
```

cada canción en el container si va añadiendo una abajo de la otra

```
<article id="list-container">
    <nav id="nav-songs">...</nav> <flex>
    <div id="song-list-content">
        <!-- Aquí van las canciones -->
        <div class="song" id="song-The Emptiness Machine">...</div> <flex>
        <div class="song" id="song-vampire">...</div> <flex>
        <div class="song" id="song-6 DE FEBRERO">...</div> <flex>
        <div class="song" id="song-UP From the bottom">...</div> <flex>
        <div class="song" id="song-Entr Nosotros (REMIX)">...</div> <flex>
        <div class="song" id="song-± DIVIDIDO">...</div> <flex>
        <div class="song" id="song-MÁS">...</div> <flex>
        <div class="song" id="song-11 RAZONES">...</div> <flex>
        <div class="song" id="song-Welcome to New York">...</div> <flex>
        <div class="song" id="song-Heavy Is The Crown">...</div> <flex>
        <div class="song" id="song-The Line">...</div> <flex>
        <div class="song" id="song-Por Si Mañana No Estoy">...</div> <flex>
        <div class="song" id="song-72 Seasons">...</div> <flex>
        <div class="song" id="song-A mí">...</div> <flex>
        <div class="song" id="song-Pantysito">...</div> <flex>
        <div class="song" id="song-Let You Fade ">...</div> <flex>
        <div class="song" id="song-Quevashacerhoy?">...</div> <flex>
        <div class="song" id="song-Un verano sin ti">...</div> <flex>
        <div class="song" id="song-Baile Inolvidable">...</div> <flex>
        <div class="song" id="song-DBTF">...</div> <flex>
    </div>
</article>
```

```
.song {
    display: flex;
    flex-direction: row;
    align-items: center;
    margin: 10px;
    padding: 10px;
    border-radius: 15px;
    background-color: var(--item-list);
    transition: ease-in .4s;
    justify-content: flex-start;
    gap: 10px;
    position: relative;
}

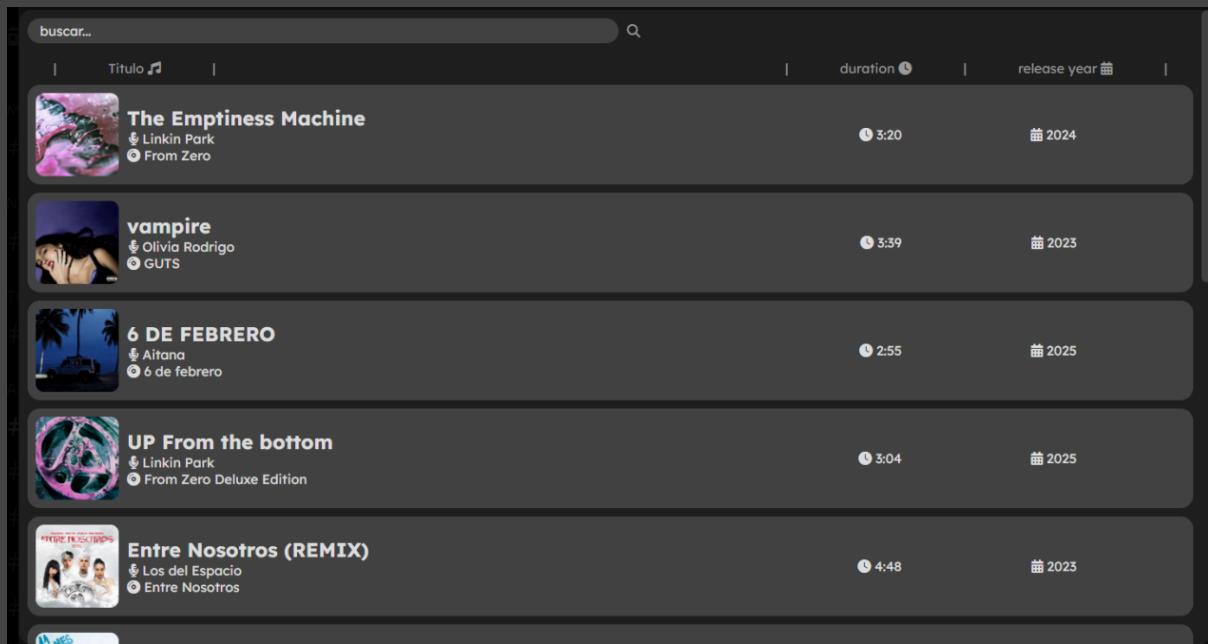
&:hover {
    background-color: var(--button-color);
    transform: scale(1.01);
    transition: ease-in 0.2s;
    cursor: pointer;
}

.song-releaseYear{
    position: absolute;
    right: 7%;
}

.song-duration{
    position: absolute;
    right: 22%;
}
```

y le añado el **css** para que se vea bien el hover lo hago con un **&:hover** que hace referencia al mismo class al que pertenece. es como una especie de **.this** para css

## Como se ve con el css e iconos



todo esto se ha estado encargando **Agustin** , pero ahora le ayudará **Pedro**  añadiendo canciones en el Json, ahora solo tenía 6 canciones en el json

## JSON CANCIONES {},

así que **Pedro**  comenzó a buscar canciones y descargar todos los mp3 y las portadas y luego va metiendo en el Json todos los datos con la estructura que cree antes: Título, artista, letra de la canción con todas las rutas etc...

```
"songs": [
    {
        "title": "Un verano sin ti",
        "artist": "Bad Bunny",
        "album": "Un verano sin ti",
        "genre": "Pop",
        "releaseYear": 2022,
        "duration": "2:29",
        "audioUrl": "/src/songs/un_verano_sin_ti.mp3",
        "albumImageUrl": "/src/img/thumbnails/un_verano_sin_ti.webp",
        "lyrics": "No sé qué pasó\nOtro amor que, de repente, fracasó\nEn mi cuarto, est"
    },
    {
        "title": "Baile Inolvidable",
        "artist": "Bad Bunny",
        "album": "Debi Tirar Mas Fotos",
        "genre": "Pop",
        "releaseYear": 2025,
        "duration": "6:18",
        "audioUrl": "/src/songs/baile_inolvidable.mp3",
        "albumImageUrl": "/src/img/thumbnails/baile_inolvidable_v_dbtf.webp"
    }
]
```

con esta forma de json  
hacemos que sea Dinámico y  
muy fácil de escalar

Y ahora solo yo Agustín  con esto:

## CARGAR CANCIONES 🎵

En la función de cargar Canciones agregamos en cada canción que generamos un Listener click

para que cuando hagas click en alguna canción de la lista seleccione una canción cambie la info de la canción y la reproduzca.

```
function loadSongs(song, songElement) {
    songElement.className = "song";
    songElement.id = `song-${song.title}`;
    songElement.innerHTML =
        `
        <div class="song-details">
            <h2>${song.title}</h2>
            <p><i class="fa-solid fa-microphone-lines" style="padding-left:2px;"></i>
                ${song.artist}</p>
            <p><i class="fa-solid fa-compact-disc"></i> ${song.album}</p>
        </div>
        <div class="song-duration">
            <p><i class="fa-solid fa-clock"></i>
                ${song.duration ? song.duration : "0:00"}</p>
        </div>
        <div class="song-releaseYear">
            <p> <i class="fa-solid fa-calendar-days"></i>
                ${song.releaseYear ? song.releaseYear : "????"}</p>
        </div>
    `;
}

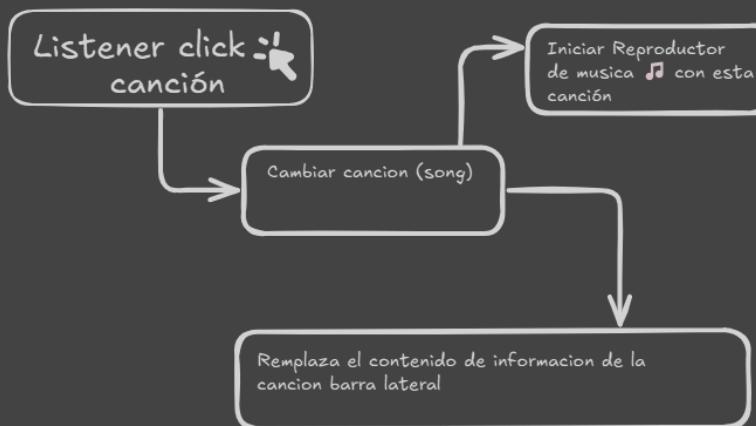
// Agregar el event listener para el clic a las canciones
songElement.addEventListener("click", function() {
    selectedSong = song;
    if (typeof changeSongInfo === 'function') {
        changeSongInfo(selectedSong);
        // Inicializar el reproductor con la URL de la canción seleccionada
        iniciarReproductor(
            selectedSong.audioUrl,
            selectedSong.title,
            selectedSong.artist,
            selectedSong.album || '',
            selectedSong.albumImageUrl
        );
    } else {
        console.error('La función changeSongInfo no está definida');
    }
})
```

la usaremos más adelante y la pongo arriba del todo :

```
// Variable global para almacenar la canción seleccionada
let selectedSong = null;
```

**explicación completa:**

```
// Agregar el event listener para el clic a las canciones
songElement.addEventListener("click", function() {
    selectedSong = song;
    if (typeof changeSongInfo === 'function') {
        changeSongInfo(selectedSong);
        // Inicializar el reproductor con la URL de la canción seleccionada
        startPlayer(
            selectedSong.audioUrl,
            selectedSong.title,
            selectedSong.artist,
            selectedSong.album || '',
            selectedSong.albumImageUrl
        );
    } else {
        console.error('La función changeSongInfo no está definida');
    }
});
```



**Si la función changeSongInfo existe:**

Llama a **changeSongInfo** para actualizar la información visual de la canción en la interfaz

Inicia el reproductor de música llamando a la función **StartPlayer** con los siguientes parámetros:

- **selectedSong.audioUrl** : La URL del archivo de audio
  - **selectedSong.title** : El título de la canción
  - **selectedSong.artist** : El nombre del artista
  - **selectedSong.album || ''** : El nombre del álbum (si no existe, usa una cadena vacía)
  - **selectedSong.albumImageUrl** : La URL de la imagen del álbum
- Si la función changeSongInfo no existe:**
- Muestra un mensaje de error en la consola

y la función `StartPlayer` la creare más adelante.

y la función `changeSongInfo` la creare a continuación:

Función `changeSongInfo` como parámetro le pasamos el objeto song para que pueda sacar sus atributos

```
//--- cambia la infomacion del reproductor
function changeSongInfo(song) {
    //---- obtengo los elementos del html
    const infoSongSide = document.getElementById("thumbnails-info-container")
    const infoSongTitle = document.getElementsByClassName("song-info-title")
    const infoSongArtist = document.getElementsByClassName("song-info-artist")
    const lyricsContainer = document.getElementById("lyrics");
    const totalTime = document.getElementById("total-time");
    //---- remplazo el contenido por el de la cancion seleccionada
    infoSongSide.src = song.albumImageUrl;
    infoSongTitle[0].textContent = song.title;
    infoSongArtist[0].textContent = song.artist;
    lyricsContainer.textContent = song.lyrics;
    totalTime.textContent = song.duration;
}
```

con los `document.getElementById()` capturó los divs h2 etc.. para cambiarle el título, artista el src a la IMG

```
//--- cambia la infomacion del reproductor
function changeSongInfo(song) {
    //---- obtengo los elementos del html
    const infoSongSide = document.getElementById("thumbnails-info-container");
    const infoSongTitle = document.getElementsByClassName("song-info-title");
    const infoSongArtist = document.getElementsByClassName("song-info-artist");
    const lyricsContainer = document.getElementById("lyrics");
    const totalTime = document.getElementById("total-time");
    //---- remplazo el contenido por el de la cancion seleccionada
    infoSongSide.src = song.albumImageUrl;
    infoSongTitle[0].textContent = song.title;
    infoSongArtist[0].textContent = song.artist;
    lyricsContainer.textContent = song.lyrics;
    totalTime.textContent = song.duration;
}
```

The Emptiness Machine  
Linkin Park

... or what I won't receive Falling for the promise  
of the emptiness machine The emptiness machine  
makes me Gave up who I was to be (like a  
sucker) I've been decided how we lose (how  
we're gonna do it) 'Cause there's a fire under the altar  
Already pulling me in Already under my skin  
And I know exactly how this ends, I Let you cut  
me open just to watch me bleed Gave up who I  
am for who you wanted me to be Don't know  
why I'm hoping for what I won't receive Falling  
for the promise of the emptiness machine



## EVENT LISTENER ALBUMS



y ahora voy a crear que cuando le des click a un álbum te oculte los álbumes y te enseñe la portada título autor del álbum y la lista de canciones pero filtrada por el álbum

```
//----- FUNCION DE ABIR EL ALBUM -----//  
//--- listener al hacer click en el album  
album1.addEventListener("click", () => {  
    clearSonglist();  
    loadAlbum(0);  
});  
album2.addEventListener("click", () => {  
    clearSonglist();  
    loadAlbum(1);  
});  
album3.addEventListener("click", () => {  
    clearSonglist();  
    loadAlbum(2);  
});  
album4.addEventListener("click", () => {  
    clearSonglist();  
    loadAlbum(3);  
});
```

estos son los listener de los 4 únicos albums  
y ahora creare La función limpiarSongList comprueba si tiene la class HIDDEN y si ya lo tiene se lo quitó y si no lo tiene lo añado

```
//----- FUNCIONES PARA OCULTAR PAGINA DE INICIO CANCIONES -----//  
function clearSonglist() {  
    const songList = document.getElementById("search-main-page");  
    if (songList.classList.contains("hidden")) {  
        // aplica el style hidden  
        songList.classList.remove("hidden");  
    } else {  
        // quita el style hidden  
        songList.classList.add("hidden");  
    }  
}  
  
function clearAlbum() {  
    const albumSongList = document.getElementById("album-song-menu");  
    if (albumSongList) {  
        // Eliminar el contenido del álbum  
        albumSongList.remove();  
    }  
}
```

y con la lista de álbumes si existe lo elimina

## Ahora la función para cargar album

```
//----- FUNCION DE ABIR EL ALBUM -----//  
//--- listener al hacer click en el album  
album1.addEventListener("click", () => {  
    clearSongList();  
    loadAlbum(0);  
});  
  
album2.addEventListener("click", () => {  
    clearSongList();  
    loadAlbum(1);  
});  
  
album3.addEventListener("click", () => {  
    clearSongList();  
    loadAlbum(2);  
});  
  
album4.addEventListener("click", () => {  
    clearSongList();  
    loadAlbum(3);  
});  
  
----- FUNCIONES PARA CARGAR ALBUM -----//  
function loadAlbum(albumNumber) {  
    const songList = document.getElementById("song-list");  
    let albumSongList = document.getElementById("album-song-menu");  
  
    // Eliminar el álbum anterior si existe  
    if (albumSongList) {  
        albumSongList.remove();  
    }  
}  
  
se pasa el indice de el album  
si existe .remove
```

en la misma función  
ahora creo un article  
le añado la id album-song-menu  
le quito la class hidden por si acaso

```
188 // -----Crear nuevo contenedor de álbum-----  
189 albumSongList = document.createElement("article");  
190 albumSongList.id = "album-song-menu";  
191 albumSongList.classList.remove("hidden"); // Asegurar que el nuevo álbum sea visible  
192  
193 albumSongList.innerHTML = `  
194     <header id="album-header">  
195           
196         <div class="album-info">  
197             <h2>${ALBUMS_LIST[albumNumber][0]}</h2>  
198             <h3>${ALBUMS_LIST[albumNumber][2]}</h3>  
199         </div>  
200         <button id="btn-back"> <i class="fa-solid fa-circle-xmark"></i> </button>  
201     </header>  
202     <nav id="nav-songs">  
203         | <span>Song <i class="fa-solid fa-music"></i></span> | <span id="separador-nav"></span> <span id="span-duration">| duration <i class="fa-solid fa-clock"></i> | <  
204     </nav>  
205 `;
```

y luego le añado con innerHTML el header del álbum  
con la información sacando de la constante de álbumes usando slicing con el  
índice que se le pasa a la función como parámetro.

```
----- CONSTANTES ALBUMS-----//  
const ALBUMS_LIST = [[{"From Zero", "/src/img/thumbnails/from-zero.webp", "Linkin Park"},  
                     ["Debi Tirar Mas Fotos", "/src/img/thumbnails/baile_inolvidable_y_dbtf.webp", "Bad Bunny"],  
                     ["11 razones", "/src/img/thumbnails/11Razones.webp", "Aitana"],  
                     ["6 de febrero", "/src/img/thumbnails/6-de-febrero.webp", "Aitana"]]];
```

y quiero que se vea así



ahora en la misma función de `loadAlbum(albumNumber)`  
sacamos de el json las canciones igual que antes pero ahora filtramos mediante

el if de que si el atributo song.album es igual a el nombre del álbum seleccionado que lo ponga en la lista

```
fetch("/src/data/data.json")
  .then((response) => response.json())
  .then((data) => {
    const songs = data.songs;
    songs.forEach((song) => {
      //ALBUMS_LIST[][][0] hacer referencia al nombre del album
      if (song.album === ALBUMS_LIST[albumNumber][0]) {
        const articleListAlbum = document.createElement("article");
        //---- añado id al elemento----
        articleListAlbum.id = "list-container-album";
        loadSongs(song, articleListAlbum);
        albumSongList.appendChild(articleListAlbum);
      }
    });
  })
  songList.appendChild(albumSongList);
```

y hago un appendChild de la AlbumSongList

para que el botón de cerrar de el album funcione creo esta función

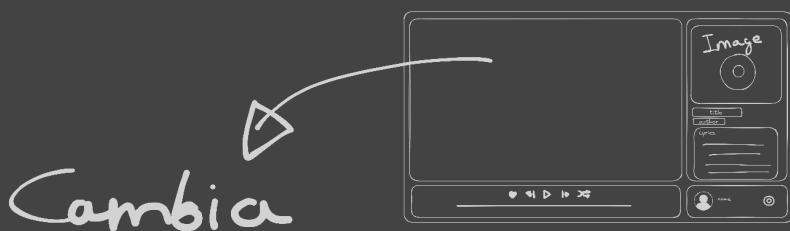
```
<button id="btn-back"> <i class="fa-solid fa-circle-xmark"></i> </button>
</header>

//----- LISTENER PARA BOTON DE VOLVER AL INICIO -----//
const btnBack = albumSongList.querySelector('#btn-back');
btnBack.addEventListener('click', () => {
  clearAlbum();
  clearSonglist();
});
```

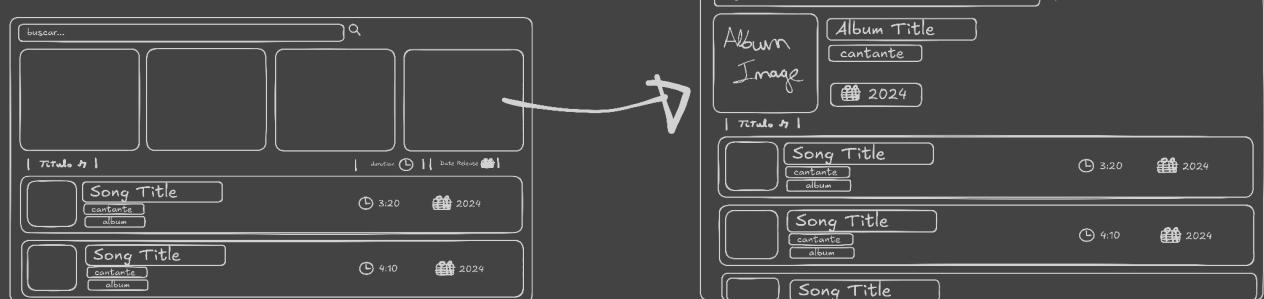
que lo que hace es limpiar el álbum

y si esta oculta la lista de canciones normal le quita el class .hidden

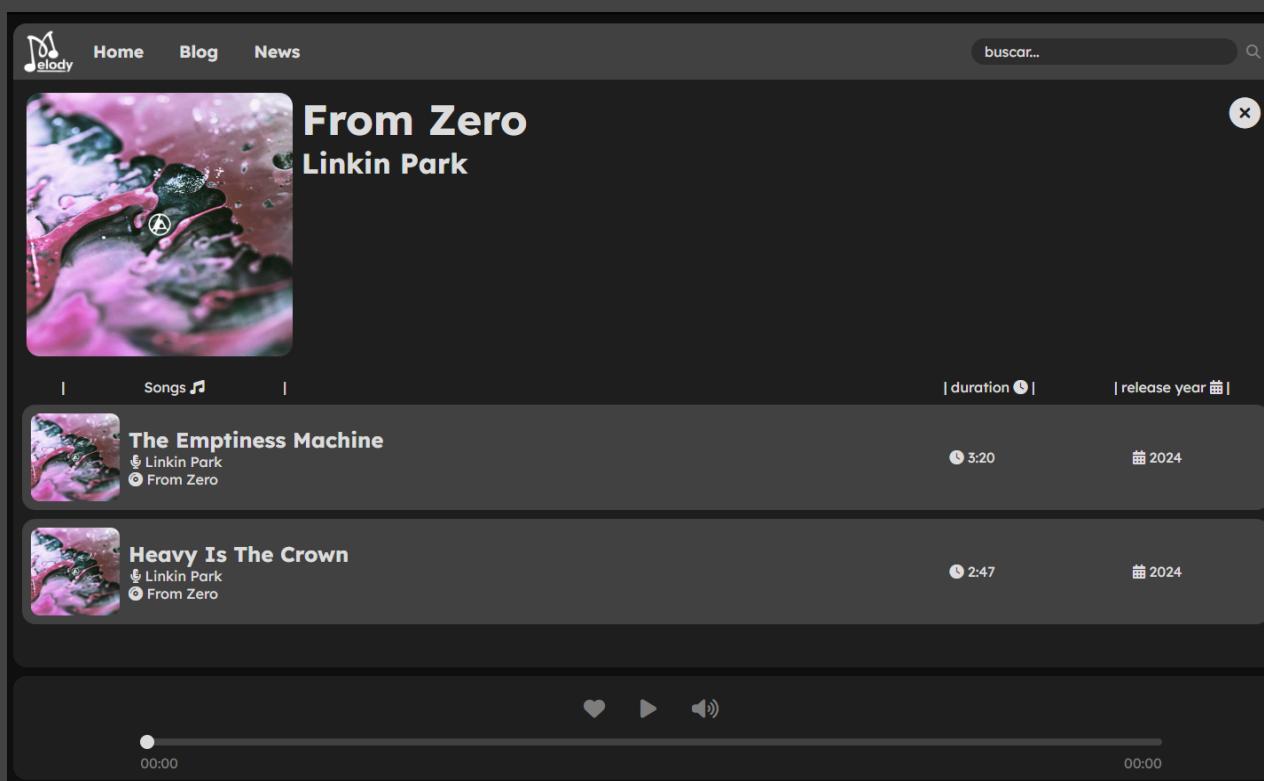
Este es el Diagrama de como funcionaria



Primero se muestra el menu con listas de reproducción



Este es el Resultado con un álbum abierto





# INPUT & FILTRAR CANCIONES

ahora con este code hago que si buscas en el input que te filtre la lista de canciones:

```
// Agregar evento de búsqueda al input
document.addEventListener("DOMContentLoaded", function() {
  const searchInput = document.querySelector("#search input");
  searchInput.addEventListener("input", function(e) {
    const searchTerm = e.target.value.toLowerCase();
    filtrarCanciones(searchTerm);
  });
});

// -----Función para filtrar canciones-----
function filtrarCanciones(searchTerm) {
  const songElements = document.querySelectorAll(".song");

  songElements.forEach(songElement => {
    const title = songElement.querySelector("h2").textContent.toLowerCase();
    const artist = songElement.querySelector(".song-details p:first-of-type").textContent.toLowerCase();
    const album = songElement.querySelector(".song-details p:last-of-type").textContent.toLowerCase();

    if (title.includes(searchTerm) ||
        artist.includes(searchTerm) ||
        album.includes(searchTerm)) {
      songElement.style.display = "flex";
    } else {
      songElement.style.display = "none";
    }
  });
}
```

le añado a el input que cuando cargue el dom que éste el eventListener que llama a filtrar canciones  
que con estas capturas del título, artista y el álbum lo paso a lower Case  
y con el queryselector le paso que sea el h2 el título con el .song-details p-first ..  
hago que el artista sea el primer p  
y el álbum sea el último p

```
const title = songElement.querySelector("h2").textContent.toLowerCase();
const artist = songElement.querySelector(".song-details p:first-of-type").textContent.toLowerCase();
const album = songElement.querySelector(".song-details p:last-of-type").textContent.toLowerCase();
```

y luego si alguno de los parámetros coincide pues lo muestra, si no lo oculta

```
if (title.includes(searchTerm) ||
    artist.includes(searchTerm) ||
    album.includes(searchTerm)) {
  songElement.style.display = "flex";
} else {
  songElement.style.display = "none";
}
```



# REPRODUCTOR DE MÚSICA (PLAYER)

y ahora voy a crear `iniciarReproductor()` y el reproductor de música antes de nada voy a crear en un archivo `player-instance.js`

Creó la function `StartPlayer()`

```
src / js / player-instance.js / ...  
1 // Crear una instancia global del reproductor  
2 const player = new Player([]);  
3  
4 // Función global para iniciar el reproductor  
5 function startPlayer(audioUrl, title = '', artist = '', album = '', imageUrl = '') {  
6   const metadata = {  
7     title:  
8     artist:  
9     album:  
10    artwork:  
11  };  
12  player.startPlayer(audioUrl, metadata);  
13}
```

Ahora creo el Archivo `Player.js`

```
class Player {  
  constructor(songList) {  
    this.songList = songList;  
    this.currentIndex = 0;  
    this.audio = new Audio();  
    this.isPlaying = false;  
    this.volume = 1.0; // Default volume  
    this.setupAudioEvents();  
    this.setupMediaSession();  
    this.setupVolumeControl();  
  }
```

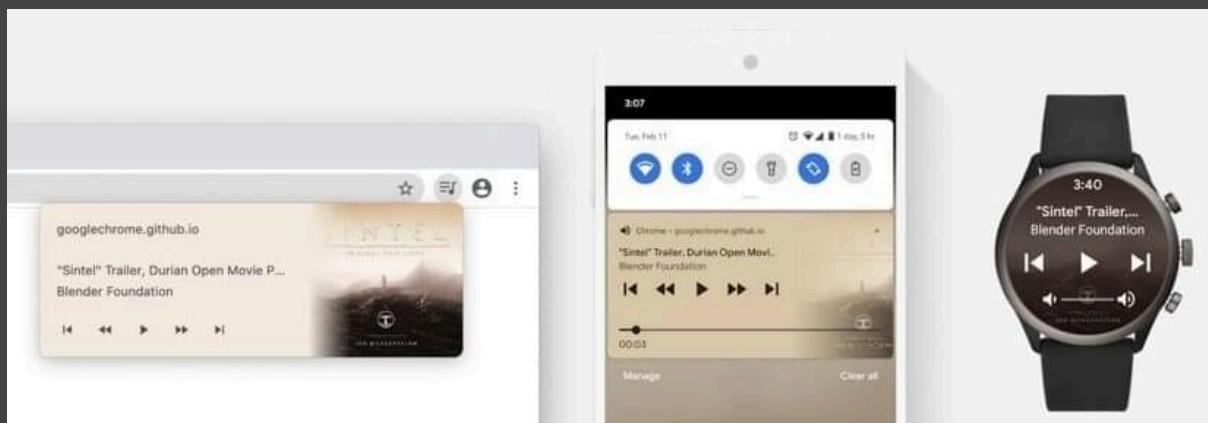
Creo la Clase Player (Reproductor)

con su constructor ; inicializo algunas cosas como el volumen a 1.0 , el isplaying false, songlist por el del parámetro del constructor.

**EN el mismo Player un método:  
de mediaSession es para usar esta API de audio de JS base**

```
//----- PARA QUE SE PUEDA CONTROLAR DESDE LAS NOTIFICACIONES DE WINDOWS & MOBILE
setupMediaSession() {
  if ('mediaSession' in navigator) {
    navigator.mediaSession.setActionHandler('play', () => this.play());
    navigator.mediaSession.setActionHandler('pause', () => this.pause());
  }
}
```

Sirve para personalizar la notificación de que se está reproduciendo la música y que se pueda pausar desde esto MediaMetaData Audio ↓



para esto también se le pasa al iniciar estos parámetros:

```
if ('mediaSession' in navigator) {
  navigator.mediaSession.metadata = new MediaMetadata({
    title: metadata.title || 'Reproduciendo música',
    artist: metadata.artist || '',
    album: metadata.album || '',
    artwork: [
      { src: metadata.artwork || '/src/img/thumbnails-thumbnails_default.webp', sizes: '512x512', type: 'image/webp' }
    ]
  });
}
```

que después se modificará dependiendo de la canción seleccionada.

sigo con la clase Player:

método para inicializar todos los eventListener  
para cuando le das en controles a play  
controlar la duración para actualizar el tiempo  
cambiar la barra de progreso de la canción (input)

```
setupAudioEvents() {  
    // Manejar el botón de control para reproducir/pausar  
    const controlButton = document.querySelector('#btn-play');  
    // Configurar el volumen inicial  
    this.audio.volume = this.volume;  
  
    // Configurar el evento timeupdate para actualizar el tiempo y la barra de progreso  
    this.audio.addEventListener('timeupdate', () => {  
        const currentTime = this.audio.currentTime;  
        const duration = this.audio.duration;  
  
        // Actualizar el tiempo actual  
        const currentTimeElement = document.querySelector('#current-time');  
        if (currentTimeElement) {  
            const minutes = Math.floor(currentTime / 60);  
            const seconds = Math.floor(currentTime % 60);  
            currentTimeElement.textContent = `${minutes}:${seconds.toString().padStart(2, '0')}`;  
        }  
  
        // Actualizar la barra de progreso  
        const progressBar = document.querySelector('#progress-bar-input');  
        if (progressBar && !isNaN(duration)) {  
            progressBar.value = (currentTime / duration) * 100;  
        }  
    });  
  
    // Agregar evento para buscar en la canción usando la barra de progreso  
    const progressBar = document.querySelector('#progress-bar-input');  
    if (progressBar) {  
        progressBar.addEventListener('input', (e) => {  
            const time = (e.target.value / 100) * this.audio.duration;  
            this.audio.currentTime = time;  
        });  
    }  
}
```

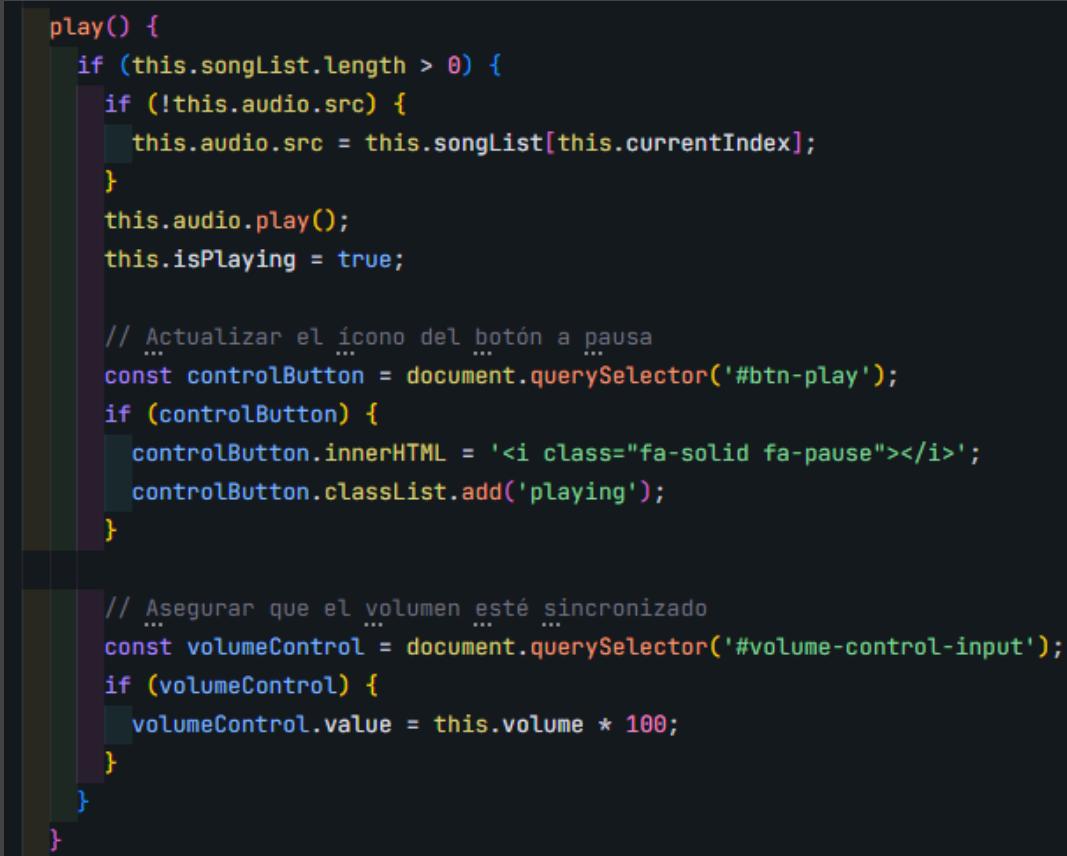


```
if (controlButton) {
  controlButton.addEventListener('click', () => {
    if (this.isPlaying) {
      this.pause();
      controlButton.innerHTML = '<i class="fa-solid fa-play"></i>';
      controlButton.classList.remove('playing');
    } else {
      this.play();
      controlButton.innerHTML = '<i class="fa-solid fa-pause"></i>';
      controlButton.classList.add('playing');
    }
  });
}
```

y para que cambie el botón de play a pause

Luego en la misma clase el metodo play()

Primero verifica si hay canciones en la lista de reproducción ( songList ). Solo procede si hay al menos una canción.



```
play() {
  if (this.songList.length > 0) {
    if (!this.audio.src) {
      this.audio.src = this.songList[this.currentIndex];
    }
    this.audio.play();
    this.isPlaying = true;

    // Actualizar el ícono del botón a pausa
    const controlButton = document.querySelector('#btn-play');
    if (controlButton) {
      controlButton.innerHTML = '<i class="fa-solid fa-pause"></i>';
      controlButton.classList.add('playing');
    }

    // Asegurar que el volumen esté sincronizado
    const volumeControl = document.querySelector('#volume-control-input');
    if (volumeControl) {
      volumeControl.value = this.volume * 100;
    }
  }
}
```

Si no hay una fuente de audio establecida, asigna la URL de la canción actual (usando el índice actual).

Inicia la reproducción del audio y marca el estado de reproducción como activo. y cambia el botón de pause y el volumen.

y pause cambia el estado de reproducción como no activo.

y para de reproducir la canción. y lo de volumen es para que no se pierda el volumen al volver a reproducir.

```

pause() {
    this.audio.pause();
    this.isPlaying = false;

    // Mantener el control de volumen sincronizado
    ...
    const volumeControl = document.querySelector('#volume-control-input');
    if (volumeControl) {
        volumeControl.value = this.volume * 100;
    }
}

```

Esta parte es también para lo que explique antes de MetaData para el mediaSession y cambia la url del audio y si está en reproducción lo pausa y luego de todo aplicado lo reproduce.

```

startPlayer(audioUrl, metadata = {}) {
    // Detener la reproducción actual si existe
    if (this.isPlaying) {
        this.pause();
    }

    this.songList = [audioUrl];
    this.currentIndex = 0;
    this.audio.src = audioUrl; // Establecer la nueva URL de audio

    if ('mediaSession' in navigator) {
        navigator.mediaSession.metadata = new MediaMetadata({
            title: metadata.title || 'Reproduciendo música',
            artist: metadata.artist || '',
            album: metadata.album || '',
            artwork: [
                { src: metadata.artwork || '/src/img/thumbnails/thumbnails_default.webp', sizes: '512x512', type: 'image/webp' }
            ]
        });
    }

    this.play();
}

```

y esto es para controlar el volumen:

```

setupVolumeControl() {
    const volumeControl = document.querySelector('#volume-control-input');
    if (volumeControl) {
        // Establecer el valor inicial
        ...
        volumeControl.value = this.volume * 100;

        // Manejar cambios en el control de volumen
        volumeControl.addEventListener('input', (e) => {
            const newVolume = e.target.value / 100;
            this.setVolume(newVolume);
        });
    }
}

setVolume(value) {
    // Asegurar que el valor esté entre 0 y 1
    this.volume = Math.max(0, Math.min(1, value));
    this.audio.volume = this.volume;

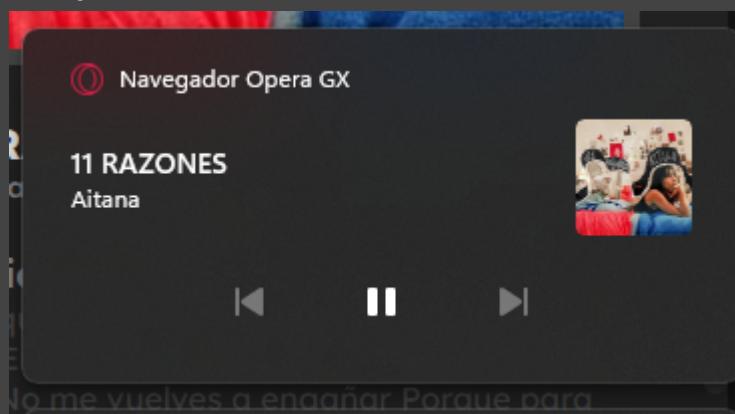
    // Actualizar el control de volumen en la interfaz
    const volumeControl = document.querySelector('#volume-control');
    if (volumeControl) {
        volumeControl.value = this.volume * 100;
    }
}

```

y como explique antes al llamar a starPlayer con estos atributos luego saldrá todo bien y se reproducirá bien

```
// Agregar el event listener para el clic a las canciones
songElement.addEventListener("click", function() {
    selectedSong = song;
    if (typeof changeSongInfo === 'function') {
        changeSongInfo(selectedSong);
        // Inicializar el reproductor con la URL de la canción seleccionada
        startPlayer(
            selectedSong.audioUrl,
            selectedSong.title,
            selectedSong.artist,
            selectedSong.album || '',
            selectedSong.albumImageUrl
        );
    } else {
        console.error('La función changeSongInfo no está definida');
    }
});
```

ejemplo:



Luego en esta misma página **main-page.html** queremos meter ajustes e inicio de sesión.

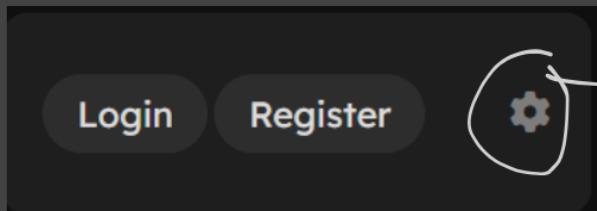
así que vamos a crear primero el ajustes **JS**



## AJUSTES | SETTINGS.JS

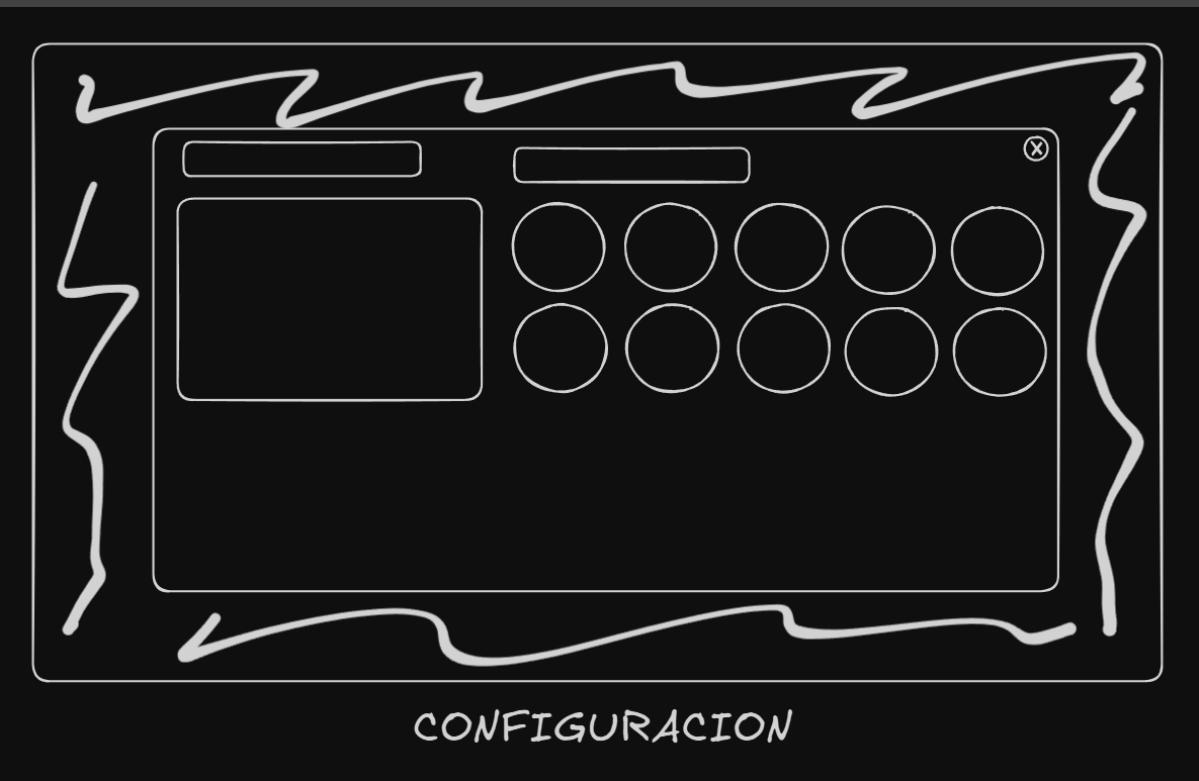
Los ajustes seran on Modal ( es una ventana emergente creada por mi )

`onclick="ocultarModal()"`



Funcion para ocultarModal()  
que quite/añada el  
class hidden al modal

El modal lo meto en el HTML de **main-page.html** directamente desde el principio no lo añado desde el **JS** .  
quiero que quede algo asi:



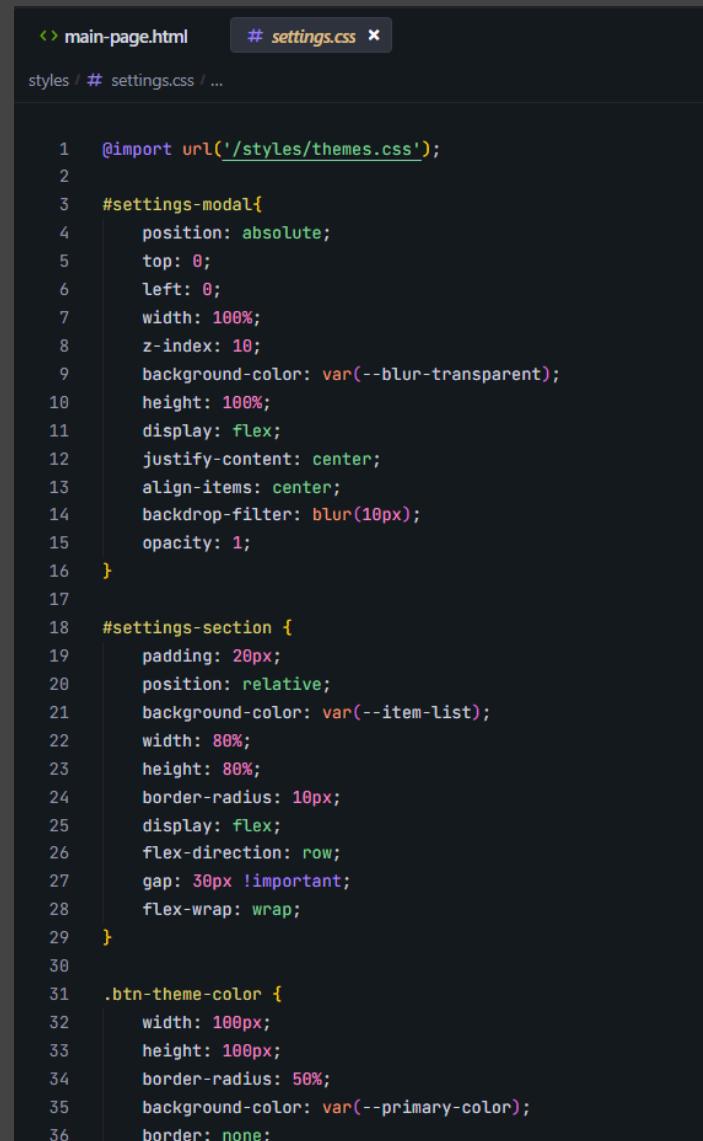
será como un popup centrado con el fondo semi transparente y BLUR (desenfoque)

El section tendrá el class hidden de base para que no se vea hasta que toque el botón :

```
<!-- Modal de Settings -->
<section id="settings-modal" class="hidden">
    <section id="settings-section">
        <button id="btn-exit-modal" onclick="ocultarModal()"><i class="fa-solid fa-circle-xmark"></i></i></button>
        <article id="appearance-containen">
            <h2>Appearance <i class="fa-solid fa-pencil"></i></h2>
            <p><i class="fa-solid fa-eye"></i> Preview of the appearance of the application.</p>
            
        </article>
        <article id="config-articles-container">
            <h3>Themes <i class="fa-solid fa-palette"></i> </h3>
            <article id="config-themes">
                <!-- Buttons de themes -->
            </article>
        </article>
    </section>
</section>
```

le añado el CSS creó el Archivo **settings.css** e importo el **themes.css** para usar las variables de los colores.

Para hacer lo del fondo hago que el section principal de settings tenga el fondo de color semi transparente con la variable (**--blur-transparent**) y le meto **backdrop-filter: blur(10px);**



The screenshot shows a code editor with two tabs: "main-page.html" and "# settings.css". The "# settings.css" tab is active, displaying the following CSS code:

```
1 @import url('/styles/themes.css');
2
3 #settings-modal{
4     position: absolute;
5     top: 0;
6     left: 0;
7     width: 100%;
8     z-index: 10;
9     background-color: var(--blur-transparent);
10    height: 100%;
11    display: flex;
12    justify-content: center;
13    align-items: center;
14    backdrop-filter: blur(10px);
15    opacity: 1;
16 }
17
18 #settings-section {
19     padding: 20px;
20     position: relative;
21     background-color: var(--item-list);
22     width: 80%;
23     height: 80%;
24     border-radius: 10px;
25     display: flex;
26     flex-direction: row;
27     gap: 30px !important;
28     flex-wrap: wrap;
29 }
30
31 .btn-theme-color {
32     width: 100px;
33     height: 100px;
34     border-radius: 50%;
35     background-color: var(--primary-color);
36     border: none;
```

esta era la IDEA <- / -> COMO QUEDÓ



para poder cambiar todos los colores de las páginas como themes diferentes  
Creo primero un CSS con todas las variables de colores para usarlo en distintos themes.

```
root {
    --primary-color: #000;
    --secondary-color: #fff;
    --background-color: #212121;
    --section-color: #212121;
    --button-color: #030303;
    --item-list: #424242;
    --buscador: #424242;
    --font-color: #f0f0f0;
    --font-secondary-color: #011818;
    --font-tertiary-color: #424242;
    --blue-transparent: #4d4d4d55;
    --salin-cancelar: #002557;
    --salin-cancelar-hover: #011818;
    /* ---fuente--- */
    --fuente-text: "Lexend", Arial, Helvetica, sans-serif;
}

.tema-default {
    --primary-color: #000;
    --secondary-color: #fff;
    --background-color: #212121;
    --section-color: #212121;
    --button-color: #030303;
    --item-list: #424242;
    --buscador: #424242;
    --font-color: #f0f0f0;
    --font-secondary-color: #011818;
    --font-tertiary-color: #424242;
    --blue-transparent: #4d4d4d55;
    --salin-cancelar: #002557;
    --salin-cancelar-hover: #011818;
}

.tema-claro {
    --primary-color: #000;
    --secondary-color: #fff;
    --background-color: #eef5f5;
    --section-color: #eef5f5;
    --button-color: #f0f0f0;
    --item-list: #e2e2e2;
    --buscador: #e2e2e2;
    --font-color: #f0f0f0;
    --font-secondary-color: #112233;
    --font-tertiary-color: #223344;
    --blue-transparent: #757575;
}

.tema-azul {
    --primary-color: #000;
    --secondary-color: #fff;
    --background-color: #637780;
    --section-color: #3f51b5;
    --button-color: #9e9eff;
    --item-list: #5cc0b8;
    --buscador: #5cc0b8;
    --font-color: #112233;
    --font-secondary-color: #9fa8da;
    --font-tertiary-color: #7986cb;
}

.tema-lila {
    --primary-color: #000;
    --secondary-color: #fff;
    --background-color: #22083a;
    --section-color: #635391;
    --button-color: #9e9eff;
    --item-list: #78517c;
    --buscador: #78517c;
    --font-color: #2e1c1f;
    --font-secondary-color: #4a344e;
    --font-tertiary-color: #7b59b6;
```

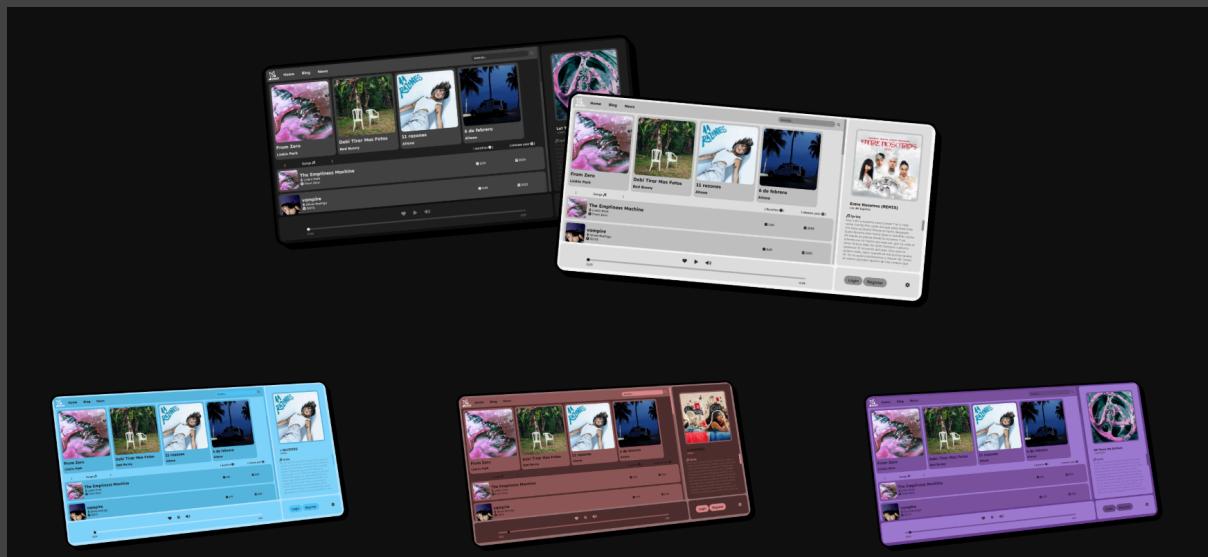
¿Cómo funciona?

defino las variables básicas  
en modo oscuro y la fuente que todos  
compartirán la misma fuente.

y para cada color se le asigna el  
mismo nombre de variables pero con  
colores diferentes

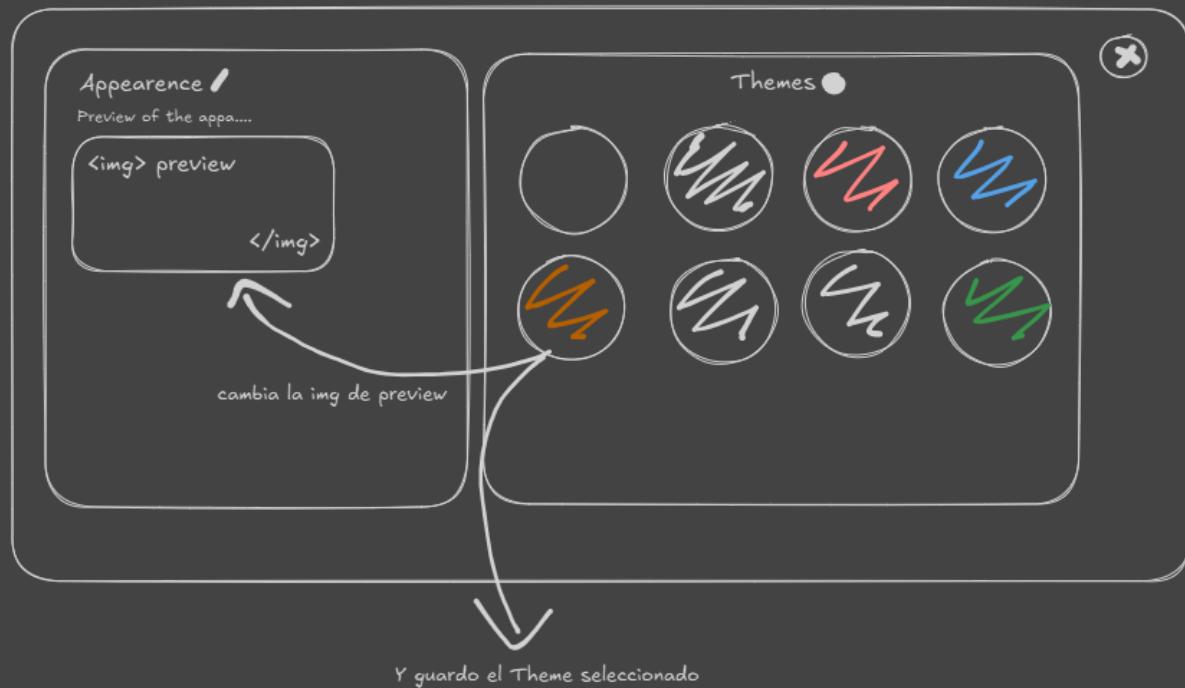
y luego esta clase .tema-claro se le  
asignará a el **body**, y cambiará el root  
por el color seleccionado.

Tenemos varios Temas de colores creados que se cambiarán en los ajustes modal



que lo añadimos fotos de cada tema en el **index.html** lo hace Marcos 

cuando cambie el tema en los ajustes lo que hará es cambiar la imagen y guardar el tema seleccionado.



JS para que Settings Funcione:

Dentro de este artículo es donde se generarán los botones para cambiar el theme

```
<article id="config-themes">  
    <!-- Buttons de themes --&gt;<br/></article>
```

estos ->



Primero en `Settings.js` voy a declarar las constantes de el número de Themes los themes que es el nombre de las clases de `themes.css`

```
//----- CONSTANTE NUMERO DE TEMAS-----//  
const NUMERO_DE_THEMES = 10;  
const THEMES = ["tema-default", "tema-claro", "tema-magenta",  
    "tema-rosa", "tema-cian", "tema-azul", "tema-marron",  
    "tema-lila", "tema-pastel", "tema-verde"];  
const THEMES_VARIABLES = "--section-color";
```

capturó el div de themes

```
// guarda el div de temas  
const themesSettings = document.getElementById('config-themes');
```

para crear los botones de themes



```
//----- CARGAR BOTONES DE TEMAS -----//  
for (let i = 0; i < NUMERO_DE_THEMES; i++) {  
    // crea un botón de tema  
    const buttonThemes = document.createElement('button');  
    buttonThemes.classList.add(THEMES[i], "btn-theme-color");  
    buttonThemes.style.backgroundColor = `var(${THEMES_VARIABLES})`;  
    buttonThemes.id = `btn-${THEMES[i]}`;  
    buttonThemes.addEventListener('click', () => {  
        changeTheme(i);  
        selecBtnTheme(i);  
    });  
    themesSettings.appendChild(buttonThemes);  
}
```

recorro con un FOR del número de temas(ahora 10)

y creo un botón en cada iteración del bucle

todos los botones tienen el mismo color definido en la constante que contiene el var `--section-color` y lo único que cambia es que en cada botón se le da el class de ese theme para que cambie el `--section-color` a el color de ese tema.

y le añado el eventListener para que cuando le haga click cambie el tema y seleccione el tema como parámetro el índice de ese color y lo añado a `themesSettings` con el `appendChild`.

**esta función cambia el theme pasado por parámetro sacando el tema como el parámetros un índice lo saca del array de la CONSTANTE.**

```
//----- FUNCION DE CAMBIAR EL TEMA -----//
function changeTheme(theme) { // 'theme' es el índice del tema en el array THEMES
    // Primero, eliminamos cualquier clase de tema existente del array THEMES del body
    THEMES.forEach(themeClassName => {
        if (document.body.classList.contains(themeClassName)) {
            document.body.classList.remove(themeClassName);
        }
    });

    // Luego, añadimos la nueva clase de tema seleccionada
    document.body.classList.add(THEMES[theme]);
    cambiarImagenPreview(theme);

    // Guardar el tema seleccionado
    saveTheme(theme);
}
```

y llamó a **SaveTheme**(y le paso el theme por parámetros);

**¿Qué es esto de SaveTheme?**

**es una forma de guardar con LocalStorage para que si haces F5 o cierras la página no se pierda el color seleccionado.**

```
//----- CARGA TEMAS -----//
// Función para obtener el tema guardado
function getStoredTheme() {
    return localStorage.getItem('selectedTheme') || 0;
}

// Función para guardar el tema seleccionado
function saveTheme(themeIndex) {
    localStorage.setItem('selectedTheme', themeIndex);
}

// Inicializar el tema al cargar la página
window.addEventListener('load', () => {
    const storedTheme = getStoredTheme();
    changeTheme(parseInt(storedTheme)); // Aplicar el tema guardado
    selecBtnTheme(parseInt(storedTheme)); // Marcar el botón del tema guardado
});
```

**Función para cambiar la imagen de la izquierda en Settings del tema selected**

```
//----- FUNCION DE CAMBIAR LA IMAGEN DE PREVIEW -----//
function cambiarImagenPreview(theme) {
    const previewImage = document.getElementById('appearance-preview');
    previewImage.src = `/src/img/main/preview-${THEMES[theme]}.png`;
}
```

Esta función es la que cambia el estilo del botón para que salga remarcado el seleccionado.

```
//----- FUNCION DE SELECCIONAR EL BOTON DE TEMA -----//
function selecBtnTheme(theme) {
    // Primero, quitamos la clase del botón anteriormente seleccionado
    const prevSelectedBtn = document.querySelector('.btn-selected-theme');
    if (prevSelectedBtn) {
        prevSelectedBtn.classList.remove('btn-selected-theme');
    }

    // Luego, añadimos la clase al nuevo botón seleccionado
    const btnTheme = document.getElementById(`btn-${THEMES[theme]}`);
    btnTheme.classList.add('btn-selected-theme');
}
```

y lo añado al html

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

igual que todos los demás

```
<script src="/src/js/settings.js"></script>
<script src="/src/js/player.js"></script>
<script src="/src/js/player-instance.js"></script>
<script src="/src/js/song-list-generator.js"></script>
```

Pedro  se encarga de crear el script de tema seleccionado en ajustes y lo carga.

```
<script src="/src/js/theme-loader.js" defer></script>
```

que se pondrán en todos los html para que cuando cargue el DOM cambie el tema.

Hola soy Pedro  y lo primero que hago es declarar la CONSTANTE de todos los temas como en el settings.js

```
js · JS theme-loader.js ⌂ applyTheme

// Constantes de temas
const THEMES = ["tema-default", "tema-claro", "tema-magenta", "tema-rosa", "tema-cian", "tema-azul", "tema-marron", "tema-lila", "tema-pastel", "tema-verde"];
```

Luego creo la función para sacar el tema seleccionado en el localStorage que ha hecho Agustín en Settings.js

```
// Función para obtener el tema guardado
function getStoredTheme() {
    return localStorage.getItem('selectedTheme') || 0;
}
```

y luego la función para aplicar el tema

```
// Función para aplicar el tema
function applyTheme() {
    const storedTheme = getStoredTheme();

    // Eliminar cualquier tema existente
    THEMES.forEach(themeClassName => {
        if (document.body.classList.contains(themeClassName)) {
            document.body.classList.remove(themeClassName);
        }
    });

    // Aplicar el tema guardado
    document.body.classList.add(THEMES[parseInt(storedTheme)]);
}

// Aplicar el tema cuando se carga la página
window.addEventListener('load', applyTheme);
```

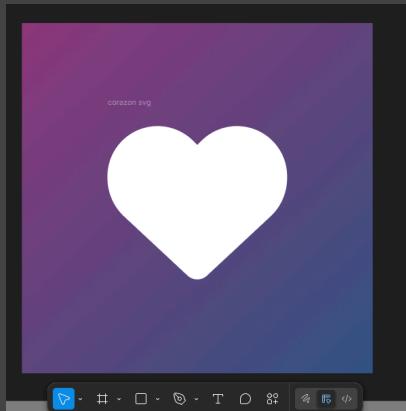
con THEMES.forEach hago que si el body del html contiene el class de esa iteración del bucle que la elimine y con el Window.addEventListener(load,applyTheme) hago que cuando cargue el documento que llame a la función applyTheme() que a la vez esa función llama a la función getStoredTheme();

Ahora este apartado sigue Agustin  el solo:

## TUS FAVORITOS

ahora en vez de haber 4 albums voy a cambiar para que sean 3 álbumes y uno al principio que sea TUS FAVORITOS y que de inicio no tenga canciones porque los demás álbumes filtran mediante el nombre del álbum pero este no este sera diferente.

creó la imagen de portada con  figma con un gradiente y un svg



Luego cambio en song-list-generator.js la CONSTANTE de álbumes

```
18 //----- CONSTANTES ALBUMS-----//  
19 const ALBUMS_LIST = [["Tus Favoritos","/src/img/thumbnails/fav.webp","",  
20 ["From Zero","/src/img/thumbnails/from-zero.webp","Linkin Park"],  
21 ["Debi Tirar Mas Fotos","/src/img/thumbnails/baile_inolvidable_y_dbtf.webp","Bad Bunny"],  
22 ["11 razones","/src/img/thumbnails/11Razones.webp","Aitana"]];  
23
```

le pongo que el nombre sea Tus Favoritos,  
la ruta la imagen creada y ningún autor

```
["Tus Favoritos","/src/img/thumbnails/fav.webp","",],
```

Luego creo un SET llamado likedSongs

```
3 let likedSongs = new Set(); // Almacena las canciones favoritas
```

para almacenar las canciones que te gustan  
ahora modiflico esta parte del code para que :

```
47 // Cargar los álbumes cuando el DOM esté listo  
48 document.addEventListener("DOMContentLoaded", () => {  
49   loadLikedSongsFromStorage(); // Cargar favoritos al iniciar  
50   loadAlbums();  
51  
52   // Agregar event listener al botón de favoritos  
53   const btnFav = document.getElementById("btn-fav");  
54   btnFav.addEventListener("click", () => {  
55     if (selectedSong) {  
56       if (likedSongs.has(selectedSong.title)) {  
57         likedSongs.delete(selectedSong.title);  
58         btnFav.querySelector("i").style.color = "";  
59       } else {  
60         likedSongs.add(selectedSong.title);  
61         btnFav.querySelector("i").style.color = "var(--magenta-btn)";  
62       }  
63       saveLikedSongsToStorage(); // Guardar cambios en localStorage  
64     }  
65   });  
66 });
```

Cargue las canciones favoritas que están guardadas en el LocalStorage

loadAlbums() no lo toco,

y añado al botón de Favoritos #btn-fav un eventListener click

que Si hay una canción seleccionada ( if (selectedSong) )

Si la canción ya está en favoritos ( likedSongs.has(selectedSong.title) ):

- La elimina de favoritos ( likedSongs.delete )

- Quita el color del ícono del corazón

Si no está en favoritos:

- La añade a favoritos ( likedSongs.add )

- Colorea el ícono del corazón en magenta usando la variable CSS --magenta-btn

Después de cualquier cambio, guarda el estado actualizado en el localStorage

con saveLikedSongsToStorage()

Ahora añado a loadAlbum un if(!window.likedSong)

Si window.likedSongs no existe Crea un nuevo conjunto ( Set ) vacío

```
//----- FUNCIONES PARA CARGAR ALBUM -----//  
function loadAlbum(albumNumber) {  
  const songList = document.getElementById("song-list");  
  let albumSongList = document.getElementById("album-song-menu");  
  
  // Eliminar el álbum anterior si existe  
  if (albumSongList) {  
    albumSongList.remove();  
  }  
  
  // Inicializar el conjunto de canciones favoritas si no existe  
  if (!window.likedSongs) {  
    window.likedSongs = new Set();  
  }
```

ahora añado en la línea 180 de song-list-generator.js

if (albumNumber === 0) Verifica si estamos en el primer álbum (Tus Favoritos)  
luego Recorro todas las canciones disponibles en el archivo data.json

```
180  if (albumNumber === 0) {  
181    // Para el álbum de favoritos, verificar si hay canciones favoritas  
182    songs.forEach((song) => {  
183      if (likedSongs.has(song.title)) {  
184        hasFavorites = true;  
185        const articleListAlbum = document.createElement("article");  
186        articleListAlbum.id = "list-container-album";  
187        loadSongs(song, articleListAlbum);  
188        albumSongList.appendChild(articleListAlbum);  
189      }  
190    });
```

if (likedSongs.has(song.title)) Verifica si el título de la canción actual está en el conjunto likedSongs

likedSongs es el Set que almacena los títulos de las canciones marcadas como favoritas

Si la canción está en favoritos:

- hasFavorites = true : Marca que hay al menos una canción favorita

Crea un nuevo elemento article para la canción

Le asigna el ID "list-container-album"

- Llama a loadSongs() para cargar la información de la canción en el elemento

- Agrega el elemento al contenedor del álbum

Iugo para si no tiene ninguna canción en favoritos genera en donde iria la lista de canciones este HTML con una lista de pasos para explicarte que tienes que hacer para añadir una canción a este álbum TUS FAVORITOS

```
// Si no hay favoritos, mostrar el tutorial
if (!hasFavorites) {
    const tutorialElement = document.createElement("div");
    tutorialElement.className = "tutorial-favorites";
    tutorialElement.innerHTML = `
        <i class="fa-solid fa-heart"></i>
        <h3>¡No tienes canciones favoritas aún!</h3>
        <p>Para agregar canciones a tus favoritos:</p>
        <ol>
            <li>Selecciona cualquier canción de la lista</li>
            <li>Haz clic en el ícono del corazón ❤ en el reproductor</li>
            <li>¡La canción aparecerá en tus favoritos!</li>
        </ol>
    `;
    albumSongList.appendChild(tutorialElement);
}
```

y añado el CSS main.css estos styles para este HTML:

```
/* ----- TUTORIAL DE FAVORITOS ----- */
.tutorial-favorites {
    text-align: center;
    padding: 20px;
    margin: 20px;
    background-color: var(--item-list);
    border-radius: 10px;
}

.tutorial-favorites i.fa-heart {
    font-size: 48px;
    color: var(--magenta-btn);
}

.tutorial-favorites h3 {
    margin-bottom: 15px;
}

.tutorial-favorites p {
    margin-bottom: 15px;
}

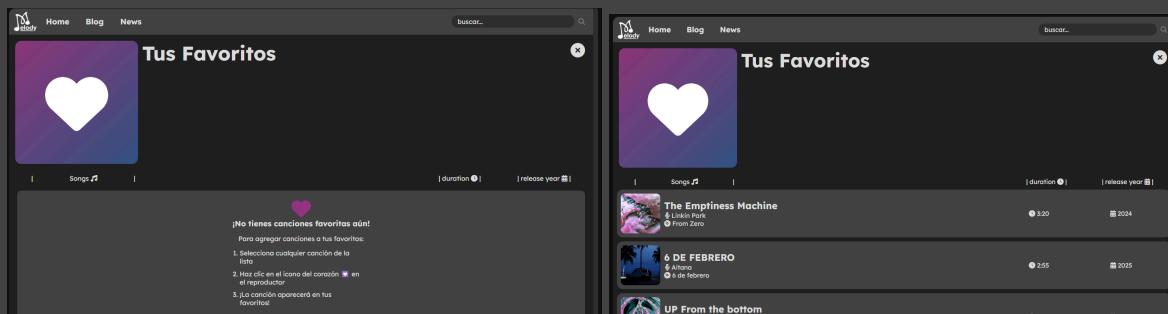
.tutorial-favorites ol {
    text-align: left;
    max-width: 300px;
    margin: 0 auto;
}

.tutorial-favorites li {
    margin-bottom: 10px;
}
```

ahora en la línea 77 de song-list-generator.js capturó el #btn-fav añado en la línea 87 cambió el estilo del botón para el icono

```
69 //--- cambia la infomacion del reproductor
70 function changeSongInfo(song) {
71     //---- obtengo los elementos del html
72     const infoSongSide = document.getElementById("thumbnails-info-container");
73     const infoSongTitle = document.getElementsByClassName("song-info-title");
74     const infoSongArtist = document.getElementsByClassName("song-info-artist");
75     const lyricsContainer = document.getElementById("lyrics");
76     const totalTime = document.getElementById("total-time");
77     const btnFav = document.getElementById("btn-fav");
78
79     //---- remplazo el contenido por el de la cancion seleccionada
80     infoSongSide.src = song.albumImageUrl;
81     infoSongTitle[0].textContent = song.title;
82     infoSongArtist[0].textContent = song.artist;
83     lyricsContainer.textContent = song.lyrics;
84     totalTime.textContent = song.duration;
85
86     // Actualizar el estado del botón de favoritos
87     if (likedSongs.has(song.title)) {
88         btnFav.querySelector("i").style.color = "var(--magenta-btn)";
89     } else {
90         btnFav.querySelector("i").style.color = "";
91     }
92 }
```

se ve así cuando no tienes fav añadidos al set:



y cuando carga alguna que le has dado sale como la lista normal pero solo salen los que les has dado a like

Y estas funciones para guardar en local storage y que no se pierda al darle F5 o salir de la web

```
// Cargar canciones favoritas desde localStorage
function loadLikedSongsFromStorage() {
    const storedSongs = localStorage.getItem('likedSongs');
    if (storedSongs) {
        likedSongs = new Set(JSON.parse(storedSongs));
    }
}

// Guardar canciones favoritas en localStorage
function saveLikedSongsToStorage() {
    localStorage.setItem('likedSongs', JSON.stringify(Array.from(likedSongs)));
}
```

aquí termina



de esta parte de JS se encargará Marcos

## LOGIN Y REGISTER

Login      Register

En el apartado de REGISTRO, usamos una Modal para poder acceder al un popup para poder registrarnos.



En este menu,podemos poner Usuario, contraseña y una foto de perfil.

-El nombre de usuario se usará para identificarte.

-La contraseña se necesitará una medianamente segura, en caso de poner una contraseña débil, te saldrá este mensaje

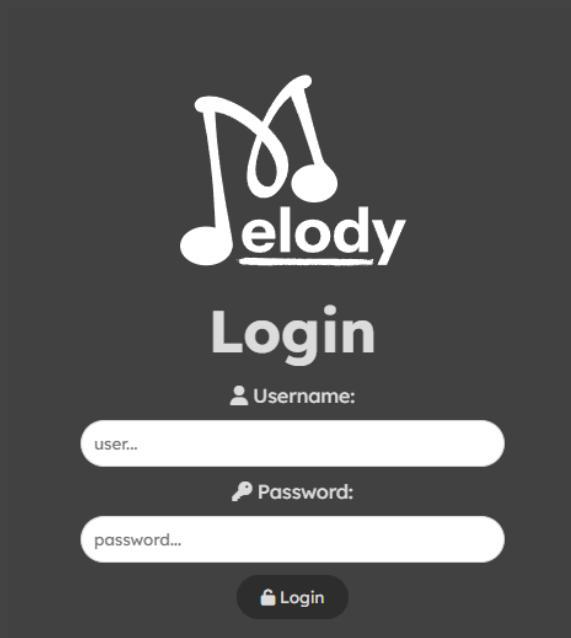
La contraseña debe tener al menos 8 caracteres, incluyendo mayúsculas, minúsculas, números y caracteres especiales (@#\$%^&\*)

```
1 // Expresiones regulares para validación del registro
2 const validations = {
3   username: /^[a-zA-Z0-9_-]{4,16}$/, // 4-16 caracteres, letras, números, guion bajo
4   password: /^(?=.*[A-Z])(?=.*[a-z])(?=.*[!@#$%^&*])[A-Za-z\d!@#$%^&*]{8,}$/ // Minimo 8 caracteres, mayúsculas, minúsculas, números y caracteres especiales
5 };
6 
```

En validations.js aplicamos esta expresión regular al nombre de usuario y la contraseña, esta expresión regular es genérica y se aplicará a todos los usuario creados. Para hacer aparecer el texto de contraseña débil o usuario mal formulado usamos la función showError().

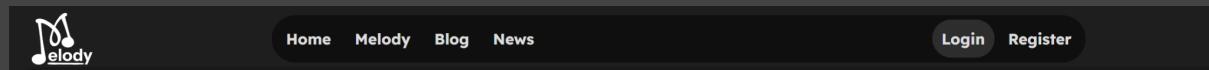
-La imagen de perfil se usará en tu perfil, su input es uno especial para subir archivos png.

Una vez ya registrado, podremos usar estas credenciales en el apartado LOGIN.



En este menú podremos poner nuestras credenciales, en caso de poner correctamente nuestro Usuario y la contraseña nos dejará acceder.

Este menú solo se usa en la pagina principal de Melody, pero en toda la web verás que hay 2 botones de Login y Register.



En caso de hacer click a estos botones te llevaran a la url /main-page.html#login en el codigo login.js hay un listener atento a la URL actual, en caso de que sea #login o #register, ejecutará la función concreta.

Si nos adentramos en el funcionamiento de login.js funciona de la siguiente forma:

**Esta función es llamada por un onclick en el html, dentro lo que hace es un inner html para mostrar el modal.**

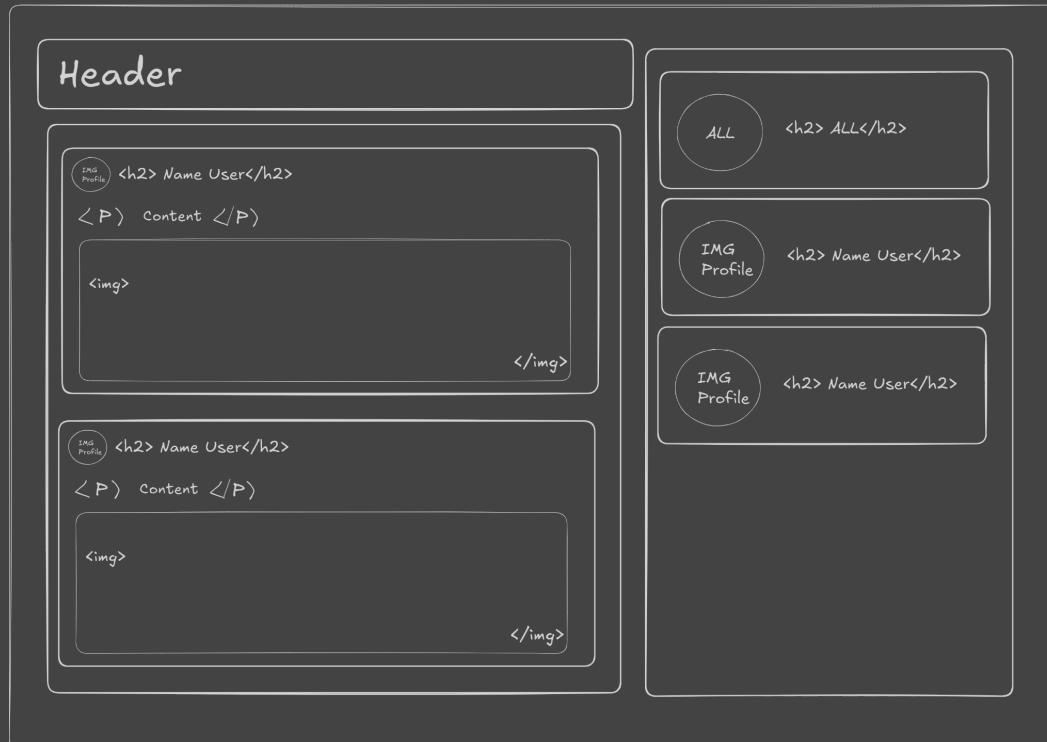
```
✓  function addRegisterHTMLModal() {
    //modal content
    loginModal.innerHTML = `
    <div class="modal-content">
        <button class="close" onclick="closeModal()"><i class="fa-regular fa-circle-xmark"></i></button>
        
        <!-- Register -->
        <h2>Register</h2>
        <form id="register-form">
            <label for="username"><i class="fa-solid fa-user"></i> Username:</label>
            <input type="text" placeholder="user..." id="register-username" name="register-username" required>
            <label for="password"><i class="fa-solid fa-key"></i> Password:</label>
            <input type="password" placeholder="password..." id="register-password" name="register-password" required>
            <label for="usericon"><i class="fa-solid fa-image"></i> Imagen de perfil:</label>
            <input type="file" id="register-usericon" name="register-usericon" accept="image/*">
            <button class="btn-submit" type="submit"><i class="fa-solid fa-floppy-disk"></i> Register</button>
        </form>
    </div>
`;
// Agregar event listener después de crear el formulario
document.getElementById('register-form').addEventListener('submit', handleRegisterSubmit);
}
```

**El login.js trabaja en conjunto con el validations.js, en caso de que queramos logearnos, de igual forma accederás por un onclick en el html.  
Después de añadir nuestras credenciales se buscará el usuario registrado.**

```
8     //----- LOGIN FORM -----//
9 ✓  function handleLoginSubmit(e) {
10    e.preventDefault();
11    const username = document.getElementById('username').value;
12    const password = document.getElementById('password').value;
13
14    // clear mensajes de error anteriores
15    formValidation.showError(document.getElementById('username'), '');
16    formValidation.showError(document.getElementById('password'), '');
17
18    // Verificar si las credenciales coinciden con alguna cuenta
19    const accountFound = accounts.find(account =>
20      account.username === username && account.password === password
21    );
22
23    if (accountFound) {
24      console.log('Login exitoso');
25      closeModal();
26      changeAccountInfoText(accountFound);
27      hiddenBtns();
28      showAccountInfoText();
29    } else {
30      console.log('Credenciales incorrectas');
31      formValidation.showError(document.getElementById('password'), 'Usuario o contraseña incorrectos');
32    }
33  }
```

# BLOG

Este es el diseño de la página:



Primero creamos el:

## Blog HTML

Usamos un header para la navegación como en otras partes de la web.

```
1  <header id="header-main">
2      <nav>
3          <div id="home-melody">
4              <a href="/index.html">Home</a>
5              <a href="/pages/main-page.html">Melody</a>
6              <a href="/pages/blog.html">Blog</a>
7              <a href="/pages/news.html">News</a>
8          </div>
9          <div class="login-register quit-mobile">
10             <a href="/pages/main-page.html#login">Login</a>
11             <a href="/pages/main-page.html#register">Register</a>
12         </div>
13
14     </nav>
15 </header>
```

**Usamos un <main> con dos <section>: el primero tiene un id para posteriormente en el JS referenciarlo y añadir los posts de los artistas y lo mismo para el segundo section, que sugerirá usuarios dentro de un <nav>.**

```
● ● ●  
1  <main>  
2      <section id="posts-container">  
3  
4          <!-- aquí van los posts -->  
5  
6      </section>  
7  
8      <section id="usuarios-sugeridos">  
9          <aside>  
10             <nav>  
11                 <!-- aquí van los usuarios -->  
12             </nav>  
13         </aside>  
14     </section>  
15 </main>
```

**En el footer, al igual que en el resto de páginas de la web, se encuentran los apartados “Sobre Nosotros”, “Contactanos” y “Términos Y Condiciones”.**

```
● ● ●  
1  <footer>  
2      <nav>  
3          <div class="privacy & about">  
4              <a href="/pages/about.html">Sobre Nosotros</a>  
5              <a href="/pages/contactar.html">Contactanos</a>  
6              <a href="/pages/terms-&-conditions.html">Terminos Y Condiciones</a>  
7          </div>  
8      </nav>  
9  </footer>
```



## Blog CSS

**Con el flex hacemos que los elementos se coloquen en fila. Además, con el gap: 2rem creamos un espacio de 2rem entre ellos.**

**Pongo el fondo transparente para que no se le aplique el color de fondo de los estilos globales que importo.**



```
1  @import url("/styles/themes.css");  
2  @import url("/styles/index.css");
```



```
1  main {  
2      display: flex;  
3      flex-direction: row;  
4      gap: 2rem;  
5      padding: 0;  
6      max-width: 1400px;  
7      margin: 0 auto;  
8      margin-top: 0;  
9      box-shadow: none;  
10     background-color: transparent;  
11 }
```

**El flex en esta porción de código permite que la foto de perfil y el nombre de usuario se alineen verticalmente al centro y se separen por un espacio de 1rem con gap.**

```
1  .post-header {  
2    display: flex;  
3    align-items: center;  
4    gap: 1rem;  
5    margin-bottom: 1rem;  
6    padding-bottom: 0.5rem;  
7    border-bottom: 1px solid var(--hover-color);  
8  }
```

**object-fit: cover** en el CSS asegura que las imágenes de perfil (de 48x48px) llenen completamente su espacio circular sin distorsionarse. Si la imagen original no es cuadrada, se recortará lo necesario para cubrir el área, garantizando que todas las fotos de perfil se vean uniformes y bien encuadradas.

```
1  .post-header .profile-pic {  
2    width: 48px;  
3    height: 48px;  
4    border-radius: 50%;  
5    object-fit: cover;  
6  }
```

**El flex hace que los botones de usuarios sugeridos se organicen en columna dentro de la barra lateral, uno debajo del otro, y gap: 1rem añade un espacio uniforme entre ellos.**

**height: fit-content** ajusta la altura de la barra lateral para que sea exactamente la necesaria para contener a todos los usuarios, sin dejar espacio extra ni cortar contenido.

**position: sticky** y **top: 2rem** hacen que la barra lateral se "pegue" a la parte superior de la pantalla (a 2rem de distancia) cuando el usuario se desplaza, manteniéndola visible mientras se exploran los posts.

```
1 #usuarios-sugeridos {  
2     width: 300px;  
3     display: flex;  
4     flex-direction: column;  
5     gap: 1rem;  
6     padding: 1.5rem;  
7     background-color: var(--background-color);  
8     border-radius: 1rem;  
9     box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);  
10    height: fit-content;  
11    position: sticky;  
12    top: 2rem;  
13 }
```

## D Blog JS

### Funcionamiento General

#### Carga Inicial de Datos:

- Con un EventListener al cargar la página el script busca y lee data-blog.json para obtener la información de todos los artistas y sus posts y la guarda en la variable allArtists
- Llama a showAllPosts() para que todas las publicaciones se muestren desde el principio.
- Crea un botón "Todos" en la barra lateral (#usuarios-sugeridos) que, al ser clickeado, vuelve a llamar a showAllPosts().
- Para cada artista, llama a sugestUsers(perfilDeArtista) para añadir su botón a la barra lateral.

```
● ● ●
```

```
1 document.addEventListener('DOMContentLoaded', function() {
2     fetch('/src/blog/data-blog.json')
3         .then(response => response.json())
4         .then(data => {
5             allArtists = data.artist;
6             showAllPosts();
7             // Crear botón "Todos"
8             const nav = document.getElementById("usuarios-sugeridos");
9             const allButton = document.createElement("button");
10            allButton.classList.add("all-button");
11            allButton.innerHTML = `<p>Todos</p>`;
12            allButton.addEventListener("click", showAllPosts);
13            nav.insertBefore(allButton, nav.firstChild);
14
15            // Añadir botones de usuarios
16            allArtists.forEach(profile => {
17                sugestUsers(profile);
18            });
19        })
20    });
});
```

## Funciones Clave

- **showAllPosts():**
  - **Primero, vacía el contenedor de posts para eliminar cualquier contenido anterior. Luego, recorre la lista allArtists y, para cada artista, llama a insertPosts(perfilDeArtista) para que se añadan sus posts.**

```
● ● ●
```

```
1 function showAllPosts() {
2     const postsContainer = document.getElementById("posts-container");
3     postsContainer.innerHTML = "";
4     allArtists.forEach(profile => {
5         insertPosts(profile);
6     });
7 }
```

- **insertPosts(profile):**
  - **Recibe un profile (objeto de artista). Recorre las publicaciones dentro de ese profile. Por cada post, crea una porción de HTML con imagen de perfil, nombre del artista, texto y, si existe, la imagen del post y lo inserta en la etiqueta con el id “posts-container”.**

```
● ○ ● ○ ●  
1 function insertPosts(profile) {  
2     const postsContainer = document.getElementById("posts-container")  
3     profile.posts.forEach(post => {  
4         const article = document.createElement("article")  
5         article.classList = "post"  
6         let postContent = `  
7             <div class="post-header">  
8                   
9                 <h3 class="user-name">${profile.name}</h3>  
10            </div>  
11            <div class="post-content">  
12                <p>${post.content}</p>  
13                ${post.media ? `` : ''}  
14            </div>  
15            article.innerHTML = postContent  
16            postsContainer.appendChild(article)  
17        })  
18    }  
19 }
```

- **changeUserProfile(profile):**

- Recibe un profile (el artista clickeado). Primero, limpia “posts-container”. Después, llama a insertPosts(profile), pero esta vez solo con los datos del artista seleccionado, mostrando únicamente sus publicaciones.

```
● ○ ● ○ ●  
1 function changeUserProfile(profile) {  
2     // Limpiar posts anteriores  
3     const postsContainer = document.getElementById("posts-container")  
4     postsContainer.innerHTML = ""  
5  
6     // Actualizar posts  
7     insertPosts(profile)  
8 }
```

- **sugestUsers(profile):**

- Recibe un profile (objeto de artista). Crea un botón HTML con la imagen de perfil y el nombre del artista. A este botón le añade un EventListener que hace que cuando el usuario haga clic en ellos, llama a changeUserProfile(profile) para filtrar los posts. Por último añade este botón a la barra lateral con el id “usuarios-sugeridos”.

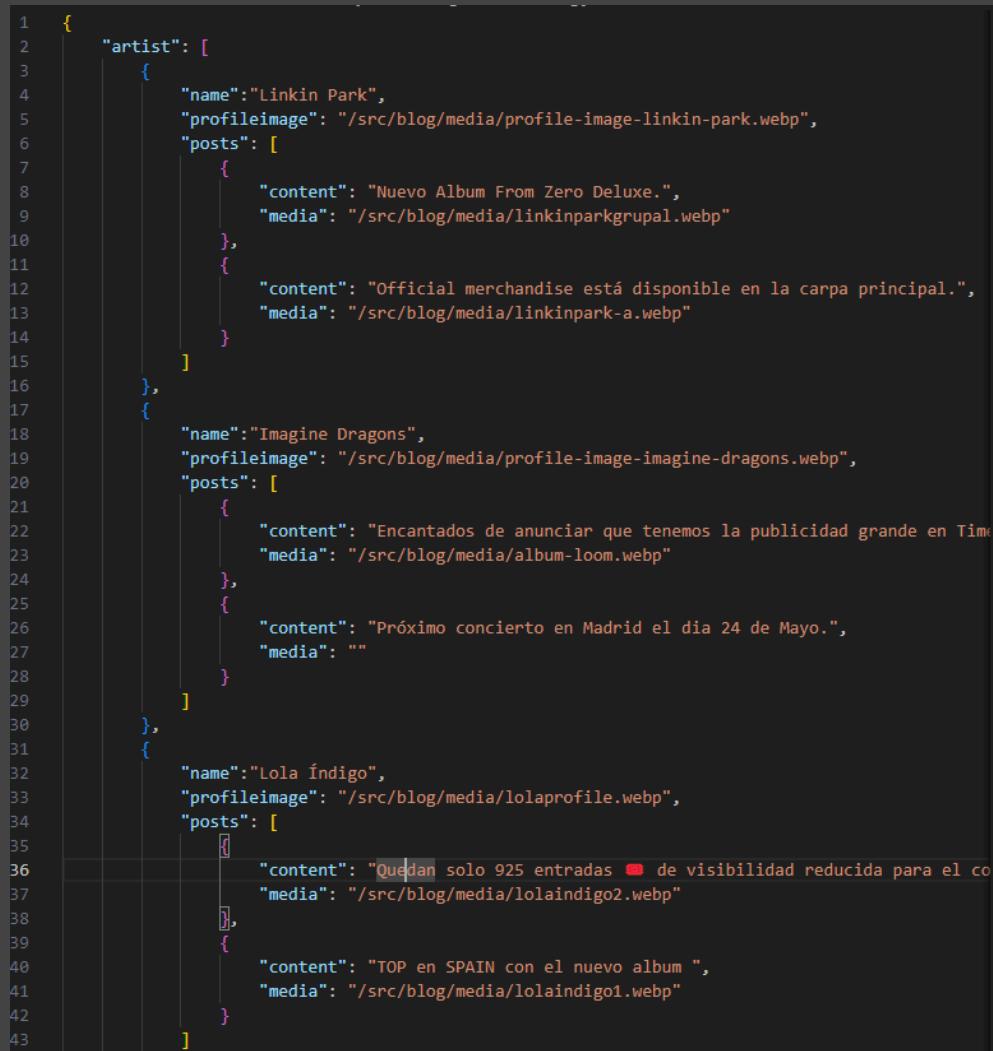


```
1 function suggestUsers(profile) {
2     const nav = document.getElementById("usuarios-sugeridos");
3     const link = document.createElement("button")
4     link.addEventListener("click", () => {
5         changeUserProfile(profile)
6     })
7     link.innerHTML = `
8     <p>${profile.name}</p>`
9     nav.appendChild(link);
10 }
```



## Blog JSON

El JSON es una lista principal de artistas. Cada artista es un objeto con su nombre, imagen de perfil, una descripción, y una lista de sus publicaciones. A su vez, cada publicación es un objeto con su contenido de texto y, opcionalmente, una imagen o video asociado.



```
1 {
2     "artist": [
3         {
4             "name": "Linkin Park",
5             "profileimage": "/src/blog/media/profile-image-linkin-park.webp",
6             "posts": [
7                 {
8                     "content": "Nuevo Album From Zero Deluxe.",
9                     "media": "/src/blog/media/linkinparkgrupal.webp"
10                },
11                {
12                    "content": "Official merchandise está disponible en la carpa principal.",
13                    "media": "/src/blog/media/linkinpark-a.webp"
14                }
15            ],
16        },
17        {
18            "name": "Imagine Dragons",
19            "profileimage": "/src/blog/media/profile-image-imagine-dragons.webp",
20            "posts": [
21                {
22                    "content": "Encantados de anunciar que tenemos la publicidad grande en Time",
23                    "media": "/src/blog/media/album-loom.webp"
24                },
25                {
26                    "content": "Próximo concierto en Madrid el dia 24 de Mayo.",
27                    "media": ""
28                }
29            ],
30        },
31        {
32            "name": "Lola Índigo",
33            "profileimage": "/src/blog/media/lolaprofile.webp",
34            "posts": [
35                [
36                    {
37                        "content": "Quedan solo 925 entradas 🚨 de visibilidad reducida para el co",
38                        "media": "/src/blog/media/lolaindigo2.webp"
39                    },
40                    {
41                        "content": "TOP en SPAIN con el nuevo album ",
42                        "media": "/src/blog/media/lolaindigo1.webp"
43                    }
44                ]
45            ]
46        }
47    ]
48 }
```

# P NEWS

Apartado que hace Pedro P

Este HTML queremos que quede así:



la estructura del html será la siguiente:

```
<body>
  <a href="/index.html">
    </a>
  <header id="header-main">
    <nav>
    </nav>
  </header>

  <main id="main-news">
    <aside id="front-news">

    </aside>
    <section id="nav-news">
      <article id="nav-news-1" class="news">

      </article>
      <article id="nav-news-2" class="news">

      </article>
      <article id="nav-news-3" class="news">

      </article>
      <article id="nav-news-4" class="news">

      </article>
    </section>
  </main>
  <footer>

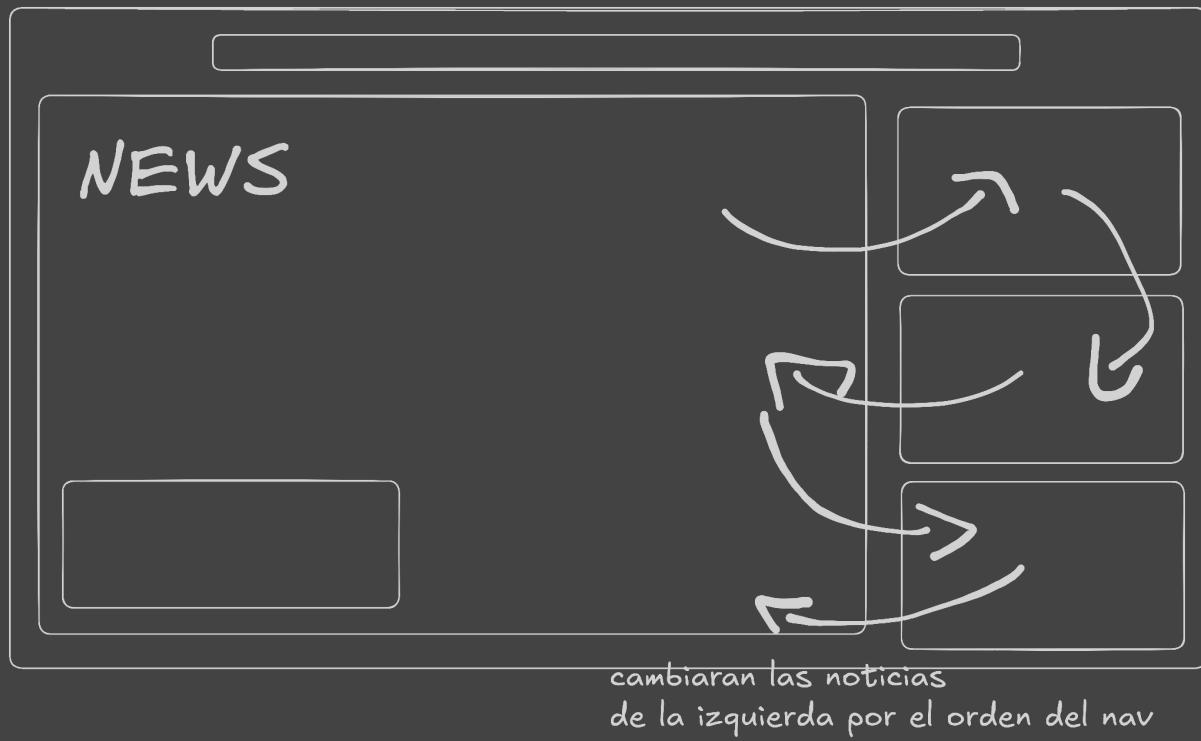
  </footer>
  <script src="/src/js/news.js"></script>
</body>
```

Luego

añado el nav de todas las otras paginas

```
<a href="/index.html">
  </a>
<header id="header-main">
  <nav>
    <div id="home-melody">
      <a href="/index.html">Home</a>
      <a href="/pages/main-page.html">Melody</a>
      <a href="/pages/blog.html">Blog</a>
      <a href="/pages/news.html">News</a>
    </div>
    <div class="login-register quit-mobile">
      <a href="">Login</a>
      <a href="">Register</a>
    </div>
  </nav>
</header>
```

La idea de esta página es que a la izquierda en el #FrontNews te muestro en grande las noticias del nav de la derecha con un slider automatico



ahora aplicare los estilos a el html con el CSS

```
<link rel="stylesheet" href="/styles/news.css">
```

también pongo el script para que cargue el Tema

```
<script src="/src/js/theme-loader.js" defer></script>
```

**Algo importante del CSS es que importo los Themes para que funcionen los colores y también los estilos básicos del Index.css para guardar la misma estética.**

```
styles / news.css / #main-description-news

1 @import url("/styles/themes.css");
2 @import url("/styles/index.css");
3
4 #main-news {
5     display: flex;
6     max-width: 1850px;
7     max-height: 90dvh;
8     flex-direction: row;
9     flex-wrap: wrap;
10    justify-content: space-between;
11 }
12
13 #image-news {
14     width: 100%;
15     height: 100%;
16     z-index: -1;
17     object-fit: cover;
18     border-radius: 10px;
19     transition: opacity 0.5s ease;
20 }
21
22 #front-news {
23     position: relative;
24     width: 1200px;
25     height: 80dvh;
26     background-color: var(--section-color);
27     border-radius: 15px;
28     margin: 10px;
29     overflow: hidden;
30 }
```

pero también meto estilos personalizados para esta página y @media screen()

Luego Creo el news.js para hacer que cambie la Front-new automatico y dependiendo a la que le des en el nav

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



## NEWS JS

Primero defino el array de newsData y el Índice de la news de ahora.

```
// -----Cargar los datos de las noticias-----
let newsData = [];
let currentNewsIndex = 0;

// -----Función para cargar los datos del JSON-----
async function loadNewsData() {
    try {
        const response = await fetch('/src/news/data-news.json');
        newsData = await response.json();
        updateMainNews(currentNewsIndex);
        generateNewsGrid();
        startNewsCarousel();
    } catch (error) {
        console.error('Error al cargar las noticias:', error);
    }
}
```

Luego creo una función asíncrona que lo que hace es que no espere a que se termine para que siga las demás ejecuciones

```
// -----Función para actualizar la noticia principal-----
function updateMainNews(index) {
    const news = newsData[index];
    const imageNews = document.getElementById('image-news');
    imageNews.style.opacity = '0';

    setTimeout(() => {
        imageNews.src = news.image;
        imageNews.style.opacity = '1';
    }, 500);

    document.getElementById('headline').textContent = news.title;
    const newsLink = document.getElementById('news-link');
    newsLink.onclick = function(e) {
        e.preventDefault();
        window.open(news.url, '_blank'); // Abre la URL en una nueva pestaña
    };
}
```

función para actualizar el front-news con una animación cada 5s

y en esta función crea en el nav las diferentes noticias y le añade un event listener

```
//----- Función para generar el Carrusel de Noticias-----
function generateNewsGrid() {
    const navNews = document.getElementById('nav-news');
    navNews.innerHTML = '';

    newsData.forEach((news, index) => {
        const article = document.createElement('article');
        article.className = 'news';
        article.id = `nav-news-${index + 1}`;

        article.innerHTML = `
            <a href="${news.url}" target="_blank">
                
                <h2>${news.title}</h2>
            </a>
        `;

        // Agregar evento click para cambiar la noticia principal
        article.addEventListener('click', (e) => {
            e.preventDefault();
            currentNewsIndex = index;
            updateMainNews(currentNewsIndex);
        });

        navNews.appendChild(article);
    });
}
```

y comienza el carrusel

```
//----- Función para iniciar el carrusel automático-----
function startNewsCarousel() {
    setInterval(() => {
        currentNewsIndex = (currentNewsIndex + 1) % newsData.length;
        updateMainNews(currentNewsIndex);
    }, 5000); // Cambiar cada 5 segundos
}

//----- Cargar las noticias cuando se carga la página-----
document.addEventListener('DOMContentLoaded', LoadNewsData);
```

y carga las noticias cuando carga el DOM



# Contribuidores | Otras Páginas

Esta página posee los contribuidores de proyectos, los usuarios involucrados serán mencionados con su nombre, imagen y todas sus redes sociales disponibles. Estos nuevos estilos serán inicialmente heredados de los estilos de la landing page, para ahorrar en texto y mantener todo más ordenado. Los menús son todos los mismos que los de la landing page.

The screenshot shows a dark-themed web page titled "Contribuidores". At the top, there is a navigation bar with links for "Home", "Melody", "Blog", "News", "Login", and "Register". Below the title, a text block states: "Este proyecto ha sido realizado por estudiantes del 1º de DAW (Desarrollo de aplicaciones Web) del IES Fernando Aguilar Quignon. Los Involucrados en el proyecto son los siguientes:". Four contributor profiles are listed in a grid:

- AgustinBenitez | @AgusTheKing**: Includes a profile picture of a person with long hair, GitHub, LinkedIn, Instagram, X (Twitter), and Own Website links.
- Trotamundos872**: Includes a profile picture of a green and white checkered pattern, GitHub, and LinkedIn links.
- The Fighter**: Includes a profile picture of a teal and white checkered pattern, GitHub, LinkedIn, and X (Twitter) links.
- Pedrux**: Includes a profile picture of a blue and white checkered pattern, GitHub, LinkedIn, and X (Twitter) links.

Para acceder a este menú, podremos acceder en el footer.

The screenshot shows a dark-themed footer with three navigation links: "Contribuidores", "Contactanos", and "Terminos Y Condiciones".



# Terminos Y Condiciones | Otras Páginas

En esta página estará dedicada a almacenar todos los derechos de la página y los derechos del usuario. Al igual que la página de contribuidores, esta tendrá el menú y los estilos inicialmente heredados de la landing page.

The screenshot shows a dark-themed website with a navigation bar at the top. The navigation bar includes links for Home, Melody, Blog, News, Login, and Register. The main content area is titled "Términos y Condiciones de Uso – Melody". Below the title, it says "Última actualización: Mayo 2025". The content is organized into numbered sections: 1. Propósito del Proyecto, 2. Uso del Contenido, 3. Limitaciones de Uso, 4. Descargo de Responsabilidad, 5. Contribuciones, and 6. Contacto. Each section contains descriptive text and bullet points where applicable. The "Uso del Contenido" section notes that Melody is an educational project. The "Limitaciones de Uso" section lists rules for use. The "Descargo de Responsabilidad" section states the project is for academic purposes. The "Contribuciones" section links to a page about contributors. The "Contacto" section provides contact information.

y una explicación básica de que no es un proyecto con ánimo de lucro si no es mero estudio y para aprender.



# Contacto | Otras Páginas

Esta página está dedicada al envío de un correo electrónico al gestor de la web, este formulario te detectará si tu email es incorrecto y te pedirá que lo cambies.

The screenshot shows a dark-themed contact form titled "Contactanos". It includes fields for "Nombre" (Name) with placeholder "Ej: Pepito de los Palotes", "Email" with placeholder "Ej: pepito@palotes.com", and a "Mensaje" (Message) area containing "Tu mensaje Aquí". A "Enviar" (Send) button is at the bottom.

Si ponemos un correo sintácticamente incorrecto, podremos ver que no nos deja mandar el correo.

The screenshot shows the same contact form but with an invalid email entry. The "Email" field contains "correo". A yellow warning box at the bottom left states: "! Incluye un signo '@' en la dirección de correo electrónico. La dirección 'correo' no incluye el signo '@'." (Include an '@' sign in the email address. The address 'correo' does not include the '@' sign.)

# BIBLIOGRAFÍA

 [Apuntes | JavaScript](#) 

 [&:hover en CSS](#)

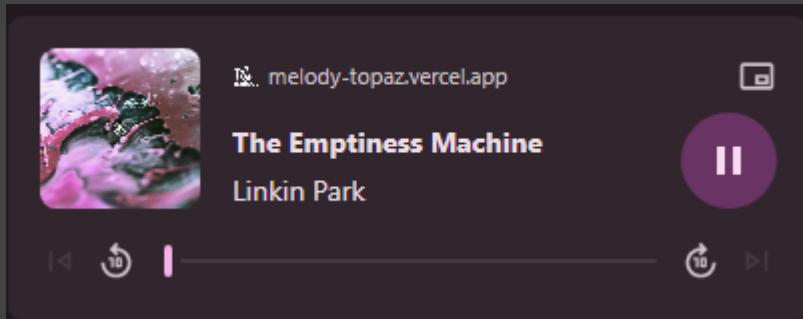
 [editar Scroll CSS](#)

 [Estilos Box Shadow](#)

 [Optimización de Imágenes WEBP](#)

 [Web Audio](#)

 [MediaMetaData Audio](#) ↓



 [Acordeones HTML](#)

 [async function](#)

 [local-storage-tutorial](#)

 [LocalStorage](#)

 [Iconos](#)

¡Gracias por ver! ❤



**Melody** | Web HOSTEADA -> <https://melody.com>