

KREITEFY

José María Martínez Franco

## INDICE

HU01 – Página de Inicio.....	2
HU02 – Registro del usuario.....	3
HU03 – Autenticación del usuario .....	4
HU04 – Página de Inicio – Novedades .....	4
HU05 – Página de Inicio – Filtros .....	5
HU06 – Ficha de una canción .....	5
HU07 – Todas las canciones .....	8
HU08 – Página de Inicio – Más escuchadas .....	11
HU09 – Página de Inicio – Para ti .....	12
HU10 – Perfil usuario .....	12
Aspecto final de la App .....	15

## HU01 – Página de Inicio

La página de Inicio la desarrollé una página simple que contaba con un navBar y el logo de la aplicación, en el navBar encontramos la opción de Log In, la cual redirige a un formulario de Inicio de sesión.

```
@PostMapping(value = "/login")
public ResponseEntity<AuthResponse> login(@RequestBody LoginDto loginDto) {
    authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(
        loginDto.getUsername(), loginDto.getPassword()));
    UserDto user = authService.getUser(loginDto.getUsername()).orElseThrow();
    String token = jwtService.generateToken(user);
    return ResponseEntity.ok(new AuthResponse(token));
}
```

```
login(): void {
    this.authService.login(this.credentials).subscribe({
        next: (response) => {
            this.errorMessage = '';
            this.token = response.token;
            this.authService.saveToken(response.token);

            // Decode token if it exists
            if (this.token) {
                this.decodedToken = this.authService.decodeToken(this.token);
                console.log('Bienvenido ' + this.decodedToken.sub); // Imprimir sub del token decodificado
            }

            this.router.navigate(['/hello']);
        },
        error: (error) => {
            console.error('Login failed', error);
            this.errorMessage = 'Error en la autenticación. Por favor, revisa tus credenciales e intenta de nuevo.';
        }
    });
}
```

Básicamente lo que he hecho es que el navBar detecte si estoy autenticado o no y me muestro Log In o Log Out según convenga y la página hello igual, así se muestra correctamente.

```
<body *ngIf="isAuthenticated">
</body>
```

```

Go to component
<!-- Navbar para usuarios autenticados -->
<nav class="barra" *ngIf="isAuthenticated">
  <div class="logo-container">
    <a [routerLink]="['']"></a>
  </div>
  <div class="links-container">
    <ul>
      <li class="nombre">
        <a [routerLink]="['../user', decodedToken?.sub]"><h2>{{ decodedToken?.sub }}</h2></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" aria-current="page" [routerLink]="['../songs']">Todas las canciones</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" aria-current="page" (click)="logout()">Log Out</a>
      </li>
    </ul>
  </div>
</nav>

<!-- Navbar para usuarios no autenticados -->
<nav class="barra" *ngIf="!isAuthenticated">
  <div class="logo-container">
    <a [routerLink]="['']"></a>
  </div>
  <div class="links-container">
    <ul>
      <li class="nav-item">
        <a class="nav-link" aria-current="page" [routerLink]="['/login']">Log In</a>
      </li>
    </ul>
  </div>
</nav>

```

## HU02 – Registro del usuario

He creado un formulario para el registro el cual cuando se completan correctamente los datos se recibe el token de usuario y ya puede entrar a la app sin necesidad de iniciar sesión.

```

@PostMapping(value = "/register")
public ResponseEntity<AuthResponse> register(@RequestBody UserDto userDto) {
    // En la base de datos no queremos guardar la contraseña, generamos
    // un hash.
    userDto.setPassword(passwordEncoder.encode(userDto.getPassword()));
    UserDto userDtoRegistered = authService.register(userDto);
    String token = jwtService.generateToken(userDtoRegistered);
    return ResponseEntity.ok(new AuthResponse(token));
}

```

```

register(): void {
    this.authService.register(this.user).subscribe({
      next: (response) => {
        this.authService.saveToken(response.token);
        this.router.navigate(['/hello']);
      },
      error: (error) => {
        console.error('There was an error!', error);
      }
    });
}

cancel(): void {
    this.router.navigate(['']);
}

```

## HU03 – Autenticación del usuario

En la autenticación del usuario la implemento gracias a JWT, la lógica se encuentra en la carpeta config.

## HU04 – Página de Inicio – Novedades

Para las novedades como ya estamos autenticados se nos muestra el contenido del body de la página hello, en la cual creamos una primera fila de canciones filtradas por lo recientes que son, lo hago creando un filtro que prioriza las canciones con id más reciente.

Además también creamos un filtro por estilo musical.

```
@GetMapping(value = @PathVariable("/songs", produces = "application/json")
public ResponseEntity<Page<SongDto>> getAllSongs(@RequestParam(value = "filter", required = false) String filter, Pageable pageable){
    Page<SongDto> songs = this.songService.getSongsByCriteriaStringPaged(pageable, filter);
    return new ResponseEntity<Page<SongDto>>(songs, HttpStatus.OK);
}
```

```
styleFilter?: string;
```

```
private getNewSongs(): void {

    const filters: string | undefined = this.buildFilters();
    this.songService.getAllSongs(this.page, this.size, this.sortById, filters).subscribe({
        next: (data: any) => {
            this.newSongs = data.content;
            this.totalPages = data.totalPages;
            this.totalElements = data.totalElements;
            this.first = data.first;
            this.last = data.last;
        },
        error: (err) => { this.handleError(err); }
    });
}
```

```

private buildFilters():string|undefined {
  const filters: string[] = [];

  if(this.styleFilter) {
    filters.push("style:MATCH:" + this.styleFilter);
  }

  if (filters.length >0) {

    let globalFilters: string = "";
    for (let filter of filters) {
      globalFilters = globalFilters + filter + ",";
    }
    globalFilters = globalFilters.substring(0, globalFilters.length-1);
    return globalFilters;

  } else {
    return undefined;
  }
}

```

## HU05 – Página de Inicio – Filtros

Implemento el filtro.

```

<div class="links-container">
  <div class = "filtro">
    <label for="style">Estilo Musical</label>
    <input type="text" [(ngModel)]="styleFilter" name="style">
  </div>

  <button class="col-12 col-md-12" (click)="limpiarCampos()">Limpiar campos</button>
  <button id="btnSearch" class="col-6 col-md-3 btn btn-secondary" (click)="searchByFilters();">Buscar</button>
</div>

```

## HU06 – Ficha de una canción

Para mostrar la información de una sola canción, creo un song-form.component, el cual através de la id de la canción que le paso, muestre toda la información sobre esta, creo un botón para generar reproducciones las cuales se van sumando al total, y un sistema de valoración de cuatro estrellas.

```

@GetMapping(value = "/songs/{songId}")
ResponseBody<SongDto> getSongById(@PathVariable Long songId) {
    Optional<SongDto> song = this.songService.getSongById(songId);

    if (song.isPresent()) {
        return new ResponseEntity<>(song.get(), HttpStatus.OK);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}

```

```

private getSongIdFromRoute(): void {
    const songIdParam = this.route.snapshot.paramMap.get('songId');
    this.songId = songIdParam ? +songIdParam : undefined;
    if (this.songId) {
        this.getSongById(this.songId);
    }
}

```

```

private getSongById(songId: number): void {
    this.songService.getSongById(songId).subscribe({
        next: (song: Song) => {
            this.song = song;
            this.currentValoration = song.valoration;
        },
        error: (error) => {
            console.error('Error fetching song:', error);
        }
    });
}

```

Para cambiar la valoración de la canción

```

changeSongValoration(): void {
  if (!this.songId || !this.newValoration) {
    console.error('Invalid song ID or new valuation.');
```

 return;
 }

 this.songService.getSongById(this.songId).subscribe({
 next: (song: Song) => {
 song.valoration = this.newValoration;
 this.songService.updateSong(song).subscribe({
 next: () => {
 console.log('Song updated successfully.');
 this.currentValoration = this.newValoration;
 },
 error: (error) => {
 console.error('Error updating song:', error);
 }
 });
 },
 error: (error) => {
 console.error('Error fetching song:', error);
 }
 });
}

Para registrar la reproducción

```

registerReproduction(): void {
  if (!this.song || !this.token) {
    console.error('Invalid song or token.');
```

 return;
 }

 this.song.reproductions = this.song.reproductions ? this.song.reproductions + 1 : 1;
 const reproduction = new Reproduction(
 10,
 this.decodedToken?.sub,
 this.song.name,
 new Date()
 );

 this.songService.updateSong(this.song).subscribe({
 next: () => {
 console.log('Reproductions registered successfully.');
 },
 error: (error) => {
 console.error('Error registering reproductions:', error);
 }
 });

 this.reproductionService.insertReproduction(reproduction).subscribe({
 next: () => {
 console.log('Reproduction registered successfully.');
 },
 error: (error) => {
 console.error('Error registering reproduction:', error);
 }
 });
}



## HU07 – Todas las canciones

He realizado un endpoint que recupere todas las canciones de la base de datos, también he creado la paginación y he creado filtros para todos los campos necesarios.

```
@GetMapping(value = @"/songs", produces = "application/json")
public ResponseEntity<Page<SongDto>> getAllSongs(@RequestParam(value = "filter", required = false) String filter, Pageable pageable){
    Page<SongDto> songs = this.songService.getSongsByCriteriaStringPaged(pageable, filter);
    return new ResponseEntity<Page<SongDto>>(songs, HttpStatus.OK);
}
```

```

export class SongListComponent implements OnInit {
  songs: Song[] = [];

  page = 0;
  size = 18;
  sort = "name,asc";

  first = false;
  last = false;
  totalPages = 0;
  totalElements = 0;

  nameFilter?: string;
  styleFilter?: string;
  artistFilter?: string;
  albumFilter?: string;

  constructor(private authService: AuthService, private songService: SongService) { }

```

```

  ngOnInit(): void {
    if (this.authService.isAuthenticated()) {
      this.getSongs();
    }
  }

  navegarSiguiente(): void {
    this.page++;
    this.getSongs();
  }

  navegarAnterior(): void {
    if (this.page > 0) {
      this.page--;
      this.getSongs();
    }
  }

  limpiarCampos(): void {
    this.nameFilter = '';
    this.styleFilter = '';
    this.artistFilter = '';
    this.albumFilter = '';
    this.searchByFilters();
  }

  searchByFilters(): void {
    this.getSongs();
  }

```

```

private buildFilters(): string | undefined {
  const filters: string[] = [];

  if (this.nameFilter) {
    filters.push("name:MATCH:" + this.nameFilter);
  }

  if (this.styleFilter) {
    filters.push("style:MATCH:" + this.styleFilter);
  }

  if (this.artistFilter) {
    filters.push("artist:MATCH:" + this.artistFilter);
  }

  if (this.albumFilter) {
    filters.push("album:MATCH:" + this.albumFilter);
  }

  if (filters.length > 0) {
    return filters.join(',');
  } else {
    return undefined;
  }
}

```

Y aquí obtengo las canciones

```

private getSongs(): void {
  const filters: string | undefined = this.buildFilters();
  this.songService.getAllSongs(this.page, this.size, this.sort, filters).subscribe({
    next: (data: any) => {
      this.songs = data.content;
      this.totalPages = data.totalPages;
      this.totalElements = data.totalElements;
      this.first = data.first;
      this.last = data.last;
    },
    error: (err) => { this.handleError(err); }
  });
}

```

```

<div class="containerFiltros">
  <form>
    <div class="links-container">
      <div class="filtro">
        <label for="name">Nombre</label>
        <input type="text" [(ngModel)]="nameFilter" name="name">
      </div>
      <div class="filtro">
        <label for="style">Estilo Musical</label>
        <input type="text" [(ngModel)]="styleFilter" name="style">
      </div>
      <div class="filtro">
        <label for="album">Album</label>
        <input type="text" [(ngModel)]="albumFilter" name="album">
      </div>
      <div class="filtro">
        <label for="artist">Artista</label>
        <input type="text" [(ngModel)]="artistFilter" name="artist">
      </div>
      <button class="col-12 col-md-12" (click)="limpiarCampos()">Limpiar campos</button>
      <button id="btnSearch" class="col-6 col-md-3 btn btn-secondary" (click)="searchByFilters();">Buscar</button>
    </div>
  </form>
</div>

```

```

<table *ngIf="songs.length > 0">
  <tbody class="song-container">
    <tr *ngFor="let song of songs">
      <td>
        <a [href]='"/songs/' + song.id"><img [src]="song.image" alt="Song Image" ></a>
        <div class="song-details">
          <h3>{{ song.name }}</h3>
        </div>
      </td>
    </tr>
  </tbody>
</table>
</div>

```

## HU08 – Página de Inicio – Más escuchadas

Volviendo a la página de inicio más escuchadas lo que he hecho es crear un nuevo método en el front que obtenga canciones, pero con otros criterios de filtrado, en este caso el número de reproducciones más alto.

Además, el filtro de estilo musical sigue afectando.

```

private getPopularSongs(): void {
  const filters: string | undefined = this.buildFilters();
  this.songService.getAllSongs(this.page, this.size, this.sortByReproductions, filters).subscribe({
    next: (data: any) => {
      this.popularSongs = data.content;
      this.totalPages = data.totalPages;
      this.totalElements = data.totalElements;
      this.first = data.first;
      this.last = data.last;
    },
    error: (err) => { this.handleError(err); }
  });
}

```

```

<h2>Las últimas novedades</h2>

<table *ngIf="newSongs.length > 0">
  <tbody class="song-container">
    <tr *ngFor="let song of newSongs">
      <td>
        <a [href]='"/songs/" + song.id"><img [src]="song.image" alt="Song Image" ></a>
        <div class="song-details">
          <h3>{{ song.name }}</h3>
        </div>
      </td>
    </tr>
  </tbody>
</table>

```

## HU09 – Página de Inicio – Para ti

```

private getForYouSongs(): void {
  const valorationFilter = 'valuation:GREATER_THAN:2';
  this.songService.getAllSongs(this.page, this.size, this.sortByReproductions, valorationFilter).subscribe({
    next: (data: any) => {
      this.forYouSongs = data.content;
      this.totalPages = data.totalPages;
      this.totalElements = data.totalElements;
      this.first = data.first;
      this.last = data.last;
    },
    error: (err) => { this.handleError(err); }
  });
}

```

## HU10 – Perfil usuario

Para acceder al Perfil de usuario debes clicar en el navBar en el nombre de usuario, allí encontrarás la página dividida en dos secciones, la de la izquierda para actualizar datos del usuario y la de la derecha para ver el historial de reproducciones del usuario logueado.

He creado un UserRestController, primero realicé un end point para obtener el user por su Id, si descodificamos el token obtenemos el id y de ahí podemos obtener todos los datos del usuario.

```

@GetMapping("/{username}")
public ResponseEntity<UserDto> getUserByUsername(@PathVariable String username) {
  UserDto user = authService.getUser(username).orElseThrow();
  return ResponseEntity.ok(user);
}

```

```

ngOnInit(): void {
  this.token = this.authService.getToken();
  if (this.token) {
    this.decodedToken = this.authService.decodeToken(this.token);
    this.isAuthenticated = true;
  }
}

```

```

getUserByUserName(username: string): Observable<any> {
  const token = this.authService.getToken();
  if (token) {
    const headers = new HttpHeaders({
      Authorization: `Bearer ${token}`
    });

    return this.http.get<any>(`${this.baseUrl}/users/${username}`, { headers });
  } else {
    console.log("No se encontró el token de autenticación");
    return throwError("No se encontró el token de autenticación");
  }
}

```

```

private getUserData(): void {
  const username = this.decodedToken.sub;
  this.userService.getUserByUserName(username).subscribe({
    next: (userData: any) => {
      console.log('Datos del usuario:', userData);
      this.userData = userData;
    },
    error: (error) => {
      console.error('Error al obtener los datos del usuario:', error);
    }
  });
}

```

```

@PatchMapping("/{username}")
public ResponseEntity<UserDto> updateUser(@RequestBody UserDto updatedUser) {
  try {
    UserDto updatedUserDto = userService.updateUser(updatedUser);
    return ResponseEntity.ok(updatedUserDto);
  } catch (IllegalArgumentException e) {
    return ResponseEntity.notFound().build();
  }
}

```

```

updateUser(): void {
  if (!this.userData) {
    return;
  }
  this.userService.updateUser(this.userData).subscribe({
    next: (updatedUser: any) => {
      console.log("Usuario actualizado:", updatedUser);
      this.router.navigate(['../hello']);
    },
    error: (error) => {
      console.error("Error al actualizar el usuario:", error);
    }
  });
}

```

Es importante implementar una codificación de la contraseña en caso de que no lo esté y en que caso de que lo esté que no se vuelva a codificar.

```

@Override
public UserDto updateUser(UserDto updatedUser) {
  Optional<User> optionalUser = persistence.find(updatedUser.getUsername());
  if (optionalUser.isPresent()) {
    User user = optionalUser.get();
    user.setFirstName(updatedUser.getFirstName());
    user.setLastName(updatedUser.getLastName());
    user.setEmail(updatedUser.getEmail());

    if (updatedUser.getPassword().equals(user.getPassword()) || passwordEncoder.matches(updatedUser.getPassword(),
      user.getPassword())) {
      user.setPassword(user.getPassword());
    } else {
      user.setPassword(passwordEncoder.encode(updatedUser.getPassword()));
    }

    user.setRole(updatedUser.getRole());

    User savedUser = persistence.save(user);
    return mapper.toDto(savedUser);
  } else {
    throw new IllegalArgumentException("Usuario no encontrado");
  }
}

```

En cuanto al historial implemento una consulta q filtre las reproducciones a partir del Nombre del usuario que las ha generado.

```

@GetMapping(value = "/reproductions/userName")
public ResponseEntity<List<ReproductionDto>> getReproductionsByUserName(@RequestParam String userName) {
  List<ReproductionDto> reproductions = this.reproductionService.getReproductionsByUserName(userName);

  if (!reproductions.isEmpty()) {
    return new ResponseEntity<>(reproductions, HttpStatus.OK);
  } else {
    return new ResponseEntity<>(HttpStatus.NOT_FOUND);
  }
}

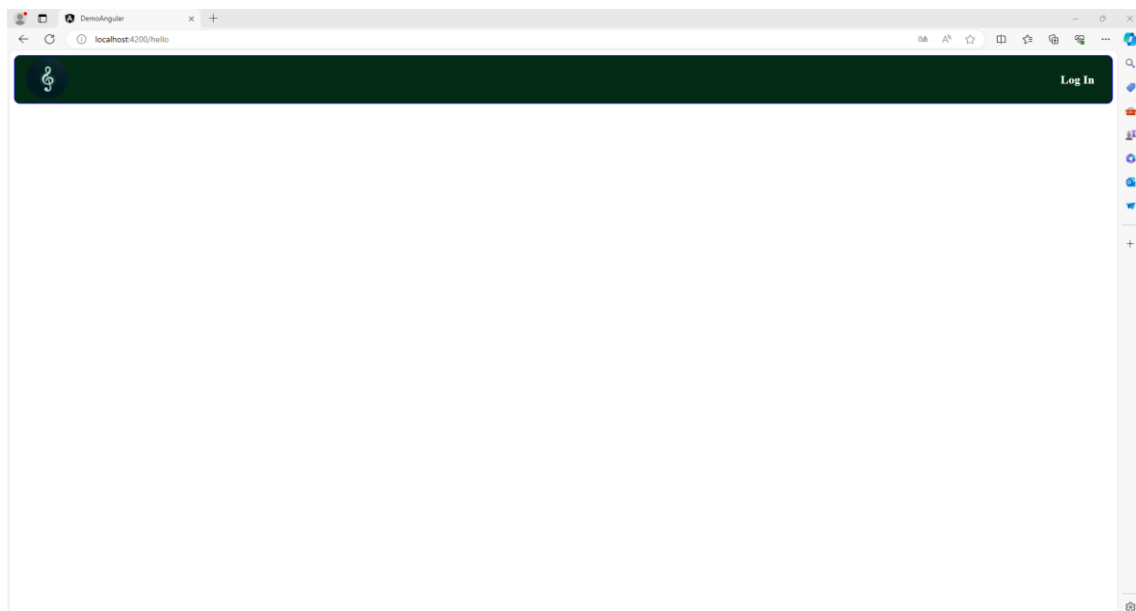
```

```
private getUserReproductions(): void {
  this.reproductionService.getReproductionsByUserName(this.decodedToken.sub, this.page, this.size, this.sort).subscribe({
    next: (reproductions: any[]) => {
      console.log('Reproducciones del usuario:', reproductions);
      this.reproductions = reproductions;
    },
    error: (error) => {
      console.error('Error al obtener las reproducciones del usuario:', error);
      console.error(this.decodedToken.sub);
    }
  });
}
```

```
public getReproductionsByUserName(userName: String, page:number, size:number, sort:string): Observable<any[]> {
  if (userName) {
    let urlEndpoint: string = `http://localhost:8080/reproductions/userName?userName=${userName}&page=${page}&size=${size}&sort=${sort}`;
    console.error(urlEndpoint);
    const token = this.authService.getToken();
    if (token) {
      const headers = new HttpHeaders({
        Authorization: `Bearer ${token}`
      });
      return this.http.get<any[]>(urlEndpoint, { headers });
    } else {
      console.error('No se encontró el token de autenticación.');
```

## Aspecto final de la App

Al iniciar la app no se nos mostrará nada más que la opción de iniciar Sesión



Una vez hecho el logueados o autenticados podemos acceder a la app.



Login



Chema

\*\*\*\*\*

Login

¿No tienes una cuenta? [Registrate](#)

Registro



First Name

Last Name

Email

Chema

\*\*\*\*\*

USER

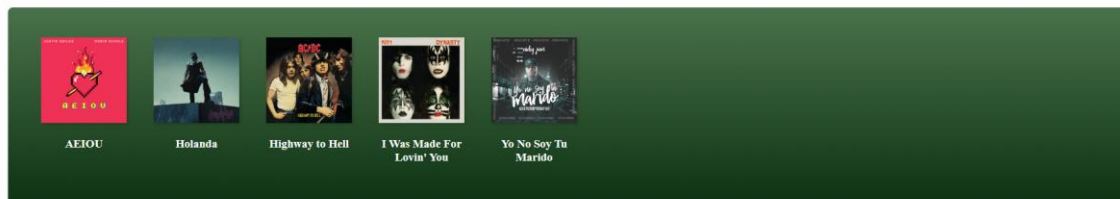
Cancelar

Registrar

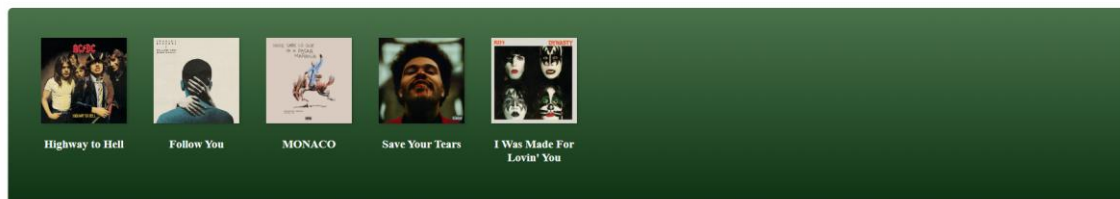
La página de inicio con las últimas novedades, las más escuchadas y el para ti, con el filtro de estilo y paginadas.

Envío Musical

Las últimas novedades



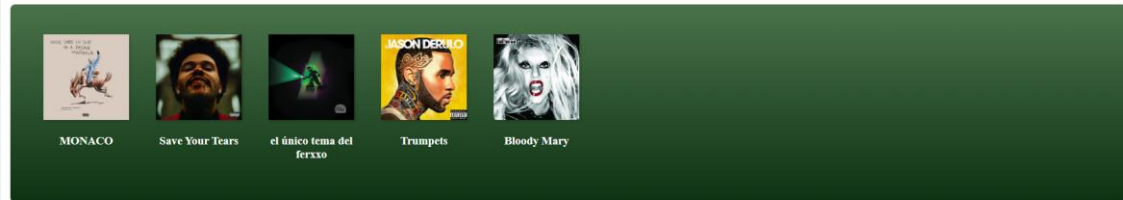
Las más escuchadas



#### Las más escuchadas



#### Para ti



Anterior

Página 1 de 4 - Elementos totales: 18

Siguiente

[Chema](#)
[Todas las canciones](#)
[Log Out](#)

#### Las últimas novedades



#### Las más escuchadas



Pulsando en el enlace de todas las canciones accedemos a una página donde podemos ver todas las canciones, paginadas y con todos los campos para filtrar.

[Chema](#)
[Todas las canciones](#)
[Log Out](#)

AEIOU

Bloody Mary

COSITAS

Everybody's changing

Fix You

Follow You

Happier Than Ever

Highway to Hell

Holanda

I Was Made For Lovin' You

Judas

MONACO

Save Your Tears

Take on Me

Trumpets

Tu jardín con enanitos


Yo No Soy Tu Marido

el único tema del ferxxo

Anterior

Página 1 de 1 - Elementos totales: 18

Siguiente



Chema

Todas las canciones

Log Out

Nombre


Estilo Musical

Album

Artista

Limpiar campos

Buscar



Save Your Tears


Anterior

Página 1 de 1 - Elementos totales: 1

Siguiente

Tanto en el inicio como en todas las canciones, podremos clicar sobre una canción y acceder a su ficha, dentro de esta podremos cambiar la valoración de la canción y reproducirla.

Volver



Save Your Tears

Style

Artist

Album

Valoración

Duration

Reproductions

Rock

The Weeknd

After Hours

4

240

190

★★★★

En cambio, si pulsamos sobre nuestro nombre de usuario en el navbar accederemos a nuestra ficha de perfil, donde podremos editar nuestros datos y visualizar nuestras reproducciones.

Volver

Nombre de Usuario

Contraseña

Nombre

Apellido

Email

Carmen

Carmen

Fernández

carmen@gmail.com

Guardar

Reproducciones del Usuario

ID de Canción	Nombre de la Canción	Fecha
11167	I Was Made For Lovin' You	2024-05-13T10:45:09.610+00:00
12752	Bloody Mary	2024-05-13T15:28:21.968+00:00
12753	Bloody Mary	2024-05-13T15:28:22.788+00:00
19152	AEIOU	2024-05-14T13:39:43.285+00:00
19153	MONACO	2024-05-14T13:39:47.077+00:00

Anterior


Página 1 de 0 - Elementos totales: 0

Siguiente


Cuando pulsemos Log Out se cerrará la sesión y deberemos autenticarnos para volver a acceder.

Estilo Musical


Las últimas novedades




AEIOU




Holanda



Highway to Hell



I Was Made For Lovin' You



Yo No Soy Tu Marido

Las más escuchadas



Highway to Hell



Follow You



MONACO



Save Your Tears



I Was Made For Lovin' You