UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

FACULTAD DE INGENIERÍA DE SISTEMAS COMPUTACIONALES

DEPARTAMENTO DE INGENIERÍA EN SOFTWARE

LICENCIATURA EN INGENIERÍA EN SOFTWARE

INGENIERÍA WEB

PROYECTO SEMESTRAL: FASE 4

DIEGO CORRALES 8-1001-1890

JUAN COPRI 8-976-1088

DANIEL PEREZ 20-14-7935

JOSUÉ PINO 8-1012-688

PROFESORA: DRA. ELBA VALDERRAMA

GRUPO 1SF134

II SEMESTRE

2025

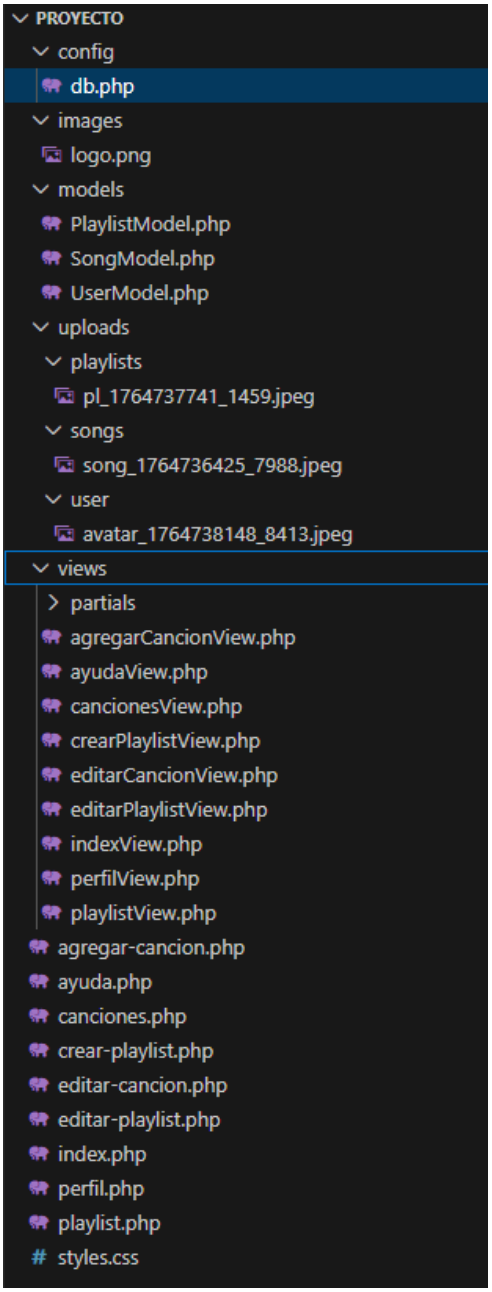
Índice

Fase 4: BackEnd

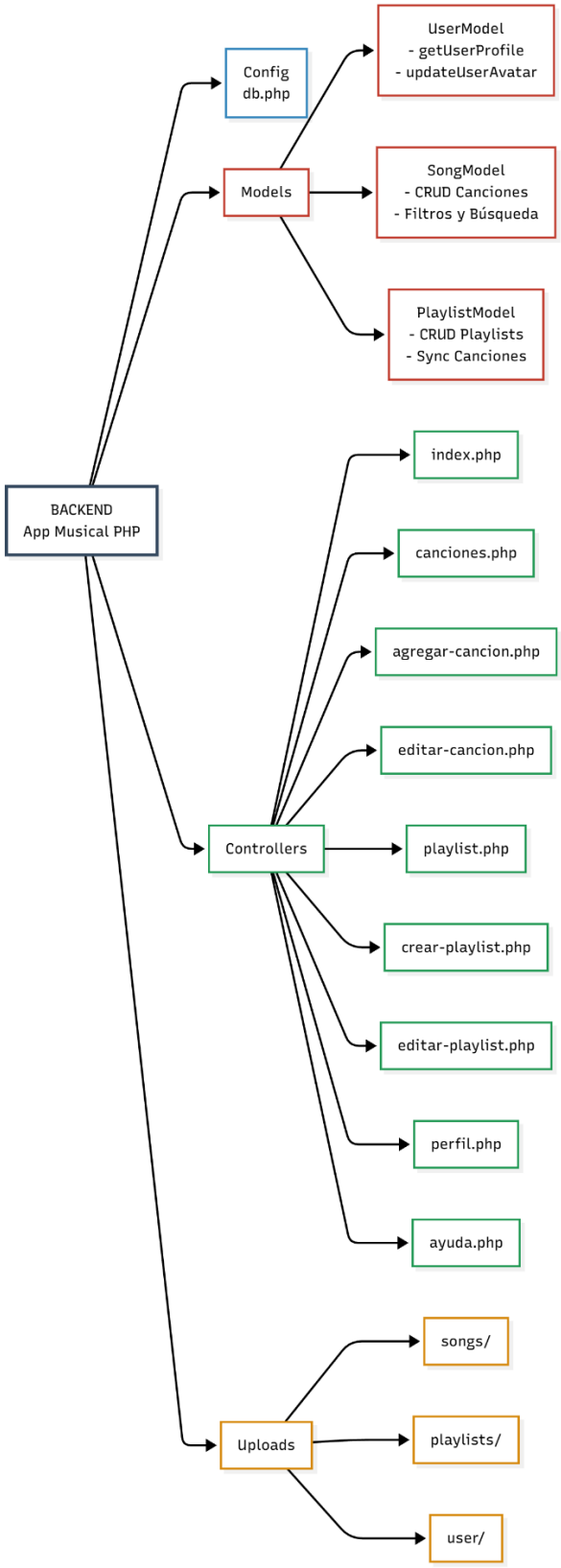
Enlace De Repositorio

Link: https://github.com/crdz05/IngWeb---ProyectoFinal-Pino-Corrales-Perez-Copri

Estructura Del BackEnd Y El Html



La Estructura Jerárquica/Diagrama



Enlace Al Sitio Web

Plataforma Usada: https://dash.infinityfree.com

Link de pag web: http://fplay.rf.gd

Código

Index.php

```php
<?php
// index.php → Página de inicio (dashboard)

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/SongModel.php';
require __DIR__ . '/models/PlaylistModel.php';

$pageTitle  = 'Inicio - App Musical';
$activePage = 'inicio';

// 1. Canciones recientes (para la sección de abajo y para el card de
"Canciones")
$recentSongs = getRecentSongs($pdo, 3); // máximo 3

// Primera canción (para la imagen grande de "Canciones")
$firstSong = $recentSongs[0] ?? null;

// 2. Playlists (para el card de "Playlist")
$playlists = getPlaylists($pdo);
$firstPlaylist = $playlists[0] ?? null;

// Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/indexView.php';
require __DIR__ . '/views/partials/footer.php';
```

Canciones.php

```php
<?php
// canciones.php → Controlador de la página "Ver Canciones"

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/SongModel.php';

$pageTitle  = 'Ver Canciones - App Musical';
$activePage = 'canciones';

// --- Eliminar canción (si viene ?eliminar=ID en la URL) ---
if (isset($_GET['eliminar']) && ctype_digit($_GET['eliminar'])) {
    $id = (int) $_GET['eliminar'];

    if ($id > 0) {
        deleteSong($pdo, $id);
        header('Location: canciones.php?msg=eliminada');
        exit;
    }
}

// --- Filtros recibidos por GET ---
$search = trim($_GET['buscar']  ?? '');
$genre  = $_GET['genero']       ?? 'Todos';
$artist = $_GET['artista']      ?? 'Todos';
$year   = $_GET['ano']          ?? 'Todos';

// Array de filtros para el modelo
$filters = [
    'search' => $search,
    'genre'  => $genre,
    'artist' => $artist,
    'year'   => $year,
];

// Obtenemos las canciones desde el modelo
$canciones = getSongs($pdo, $filters);

// Listas para los combos de Artista y Año
$artistsList = getDistinctArtists($pdo);
$yearsList   = getDistinctYears($pdo);

// Mensaje opcional (ej: después de eliminar o agregar/editar)
```

```php
$message = '';
if (!empty($_GET['msg'])) {
    switch ($_GET['msg']) {
        case 'eliminada':
            $message = 'Canción eliminada correctamente.';
            break;
        case 'agregada':
            $message = 'Canción agregada correctamente.';
            break;
        case 'editada':
            $message = 'Canción actualizada correctamente.';
            break;
    }
}

// Cargamos vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/cancionesView.php';
require __DIR__ . '/views/partials/footer.php';
```

agregar-cancion.php

```php
<?php
// agregar-cancion.php → Controlador para agregar nuevas canciones

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/SongModel.php';

$pageTitle  = 'Agregar Canción - App Musical';
$activePage = 'canciones';

$errors  = [];
$oldData = [
    'titulo'   => '',
    'artista'  => '',
    'genero'   => '',
    'ano'      => '',
    'album'    => '',
    'duracion' => '',
];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // 1. Recibir datos del formulario
```

```
    $oldData['titulo']   = trim($_POST['titulo']   ?? '');
    $oldData['artista']  = trim($_POST['artista']  ?? '');
    $oldData['genero']   = trim($_POST['genero']   ?? '');
    $oldData['ano']      = trim($_POST['ano']      ?? '');
    $oldData['album']    = trim($_POST['album']    ?? '');
    $oldData['duracion'] = trim($_POST['duracion'] ?? '');

    // 2. Validaciones básicas
    if ($oldData['titulo'] === '') {
        $errors[] = 'El título de la canción es obligatorio.';
    }

    if ($oldData['artista'] === '') {
        $errors[] = 'El artista es obligatorio.';
    }

    if ($oldData['ano'] !== '' && !ctype_digit($oldData['ano'])) {
        $errors[] = 'El año debe ser un número entero.';
    }

    // 3. Manejo de la imagen (opcional)
    $imagePath = null;

    if (!empty($_FILES['imagen']['name']) && $_FILES['imagen']['error'] !==
UPLOAD_ERR_NO_FILE) {
        if ($_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
            $uploadDir = __DIR__ . '/uploads/songs/';
            if (!is_dir($uploadDir)) {
                mkdir($uploadDir, 0777, true);
            }

            $ext = strtolower(pathinfo($_FILES['imagen']['name'],
PATHINFO_EXTENSION));
            $allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];

            if (!in_array($ext, $allowed)) {
                $errors[] = 'Formato de imagen no permitido. Usa JPG, PNG,
GIF o WEBP.';
            } else {
                $basename = 'song_' . time() . '_' . mt_rand(1000, 9999) .
'.' . $ext;
                $destPath = $uploadDir . $basename;
                $webPath  = 'uploads/songs/' . $basename; // ruta que se
guarda en la BD
```

```
                if (move_uploaded_file($_FILES['imagen']['tmp_name'],
$destPath)) {
                    $imagePath = $webPath;
                } else {
                    $errors[] = 'No se pudo subir la imagen de la canción.';
                }
            }
        } else {
            $errors[] = 'Ocurrió un error al subir la imagen de la
canción.';
        }
    }

    // 4. Si no hay errores, guardar en la BD
    if (empty($errors)) {
        $data = [
            'title'      => $oldData['titulo'],
            'artist'     => $oldData['artista'],
            'genre'      => $oldData['genero'] !== '' ? $oldData['genero'] :
null,
            'year'       => $oldData['ano'] !== '' ? (int)$oldData['ano'] :
null,
            'album'      => $oldData['album'] !== '' ? $oldData['album'] :
null,
            'duration'   => $oldData['duracion'] !== '' ?
$oldData['duracion'] : null,
            'image_path' => $imagePath,
        ];

        if (createSong($pdo, $data)) {
            header('Location: canciones.php?msg=agregada');
            exit;
        } else {
            $errors[] = 'Ocurrió un error al guardar la canción en la base
de datos.';
        }
    }
}

// Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/agregarCancionView.php';
require __DIR__ . '/views/partials/footer.php';
```

editar-cancion.php

```php
<?php
// editar-cancion.php → Controlador para editar una canción

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/SongModel.php';

$pageTitle  = 'Editar Canción - App Musical';
$activePage = 'canciones';

$errors   = [];
$formData = [
    'titulo'   => '',
    'artista'  => '',
    'genero'   => '',
    'ano'      => '',
    'album'    => '',
    'duracion' => '',
];

// 1. Validar ID
if (empty($_GET['id']) || !ctype_digit($_GET['id'])) {
    header('Location: canciones.php');
    exit;
}

$id = (int) $_GET['id'];

// 2. Obtener canción de la BD
$song = getSongById($pdo, $id);

if (!$song) {
    die("La canción con ID $id no existe.");
}

// imagen actual (puede ser null)
$currentImagePath = $song['image_path'] ?? null;

// 3. Rellenar formulario inicial con datos de la canción
$formData['titulo']   = $song['title']    ?? '';
```

```php
$formData['artista']  = $song['artist']   ?? '';
$formData['genero']   = $song['genre']    ?? '';
$formData['ano']      = $song['year']     ?? '';
$formData['album']    = $song['album']    ?? '';
$formData['duracion'] = $song['duration'] ?? '';

// 4. Procesar POST (si el usuario envía el formulario)
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    // Sobrescribir con lo que viene del form
    $formData['titulo']   = trim($_POST['titulo']   ?? '');
    $formData['artista']  = trim($_POST['artista']  ?? '');
    $formData['genero']   = trim($_POST['genero']   ?? '');
    $formData['ano']      = trim($_POST['ano']      ?? '');
    $formData['album']    = trim($_POST['album']    ?? '');
    $formData['duracion'] = trim($_POST['duracion'] ?? '');

    // Validaciones básicas
    if ($formData['titulo'] === '') {
        $errors[] = 'El título de la canción es obligatorio.';
    }

    if ($formData['artista'] === '') {
        $errors[] = 'El artista es obligatorio.';
    }

    if ($formData['ano'] !== '' && !ctype_digit($formData['ano'])) {
        $errors[] = 'El año debe ser un número entero.';
    }

    // Manejo de la imagen (opcional)
    // Por defecto mantenemos la imagen actual
    $imagePath = $currentImagePath;

    if (!empty($_FILES['imagen']['name']) && $_FILES['imagen']['error'] !==
UPLOAD_ERR_NO_FILE) {
        if ($_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
            $uploadDir = __DIR__ . '/uploads/songs/';
            if (!is_dir($uploadDir)) {
                mkdir($uploadDir, 0777, true);
            }
```

```
            $ext = strtolower(pathinfo($_FILES['imagen']['name'],
PATHINFO_EXTENSION));
            $allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];

            if (!in_array($ext, $allowed)) {
                $errors[] = 'Formato de imagen no permitido. Usa JPG, PNG,
GIF o WEBP.';
            } else {
                $basename = 'song_' . time() . '_' . mt_rand(1000, 9999) .
'.' . $ext;
                $destPath = $uploadDir . $basename;
                $webPath  = 'uploads/songs/' . $basename;

                if (move_uploaded_file($_FILES['imagen']['tmp_name'],
$destPath)) {
                    $imagePath = $webPath;
                } else {
                    $errors[] = 'No se pudo subir la nueva imagen de la
canción.';
                }
            }
        } else {
            $errors[] = 'Ocurrió un error al subir la imagen de la
canción.';
        }
    }

    // Si no hay errores, actualizar en la BD
    if (empty($errors)) {
        $data = [
            'title'      => $formData['titulo'],
            'artist'     => $formData['artista'],
            'genre'      => $formData['genero'] !== '' ? $formData['genero']
: null,
            'year'       => $formData['ano'] !== '' ? (int)$formData['ano']
: null,
            'album'      => $formData['album'] !== '' ? $formData['album'] :
null,
            'duration'   => $formData['duracion'] !== '' ?
$formData['duracion'] : null,
            'image_path' => $imagePath,
        ];

        if (updateSong($pdo, $id, $data)) {
```

```
            header('Location: canciones.php?msg=editada');
            exit;
        } else {
            $errors[] = 'Ocurrió un error al actualizar la canción en la
base de datos.';
        }
    }
}

// 5. Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/editarCancionView.php';
require __DIR__ . '/views/partials/footer.php';
```

Playlist.php

```
<?php
// playlist.php → Controlador para "Mis Playlists"

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/PlaylistModel.php';

$pageTitle  = 'Mis Playlists - App Musical';
$activePage = 'playlist';

$message = '';

// Eliminar playlist si viene ?eliminar=ID
if (isset($_GET['eliminar']) && ctype_digit($_GET['eliminar'])) {
    $id = (int) $_GET['eliminar'];

    if ($id > 0) {
        if (deletePlaylist($pdo, $id)) {
            header('Location: playlist.php?msg=eliminada');
            exit;
        } else {
            $message = 'No se pudo eliminar la playlist. Intente
nuevamente.';
        }
    }
}

// Mensaje opcional por acciones anteriores
```

```php
if (isset($_GET['msg'])) {
    if ($_GET['msg'] === 'creada') {
        $message = 'Playlist creada correctamente.';
    } elseif ($_GET['msg'] === 'editada') {
        $message = 'Playlist actualizada correctamente.';
    } elseif ($_GET['msg'] === 'eliminada') {
        $message = 'Playlist eliminada correctamente.';
    }
}

// Obtener playlists desde el modelo
$playlists = getPlaylists($pdo);

// Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/playlistView.php';
require __DIR__ . '/views/partials/footer.php';
```

crear-playlist.php

```php
<?php
// crear-playlist.php → Controlador para crear una nueva playlist

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/PlaylistModel.php';

$pageTitle  = 'Crear Playlist - App Musical';
$activePage = 'playlist';

$errors   = [];
$formData = [
    'nombre'      => '',
    'descripcion' => '',
];

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // 1. Datos del formulario
    $formData['nombre']      = trim($_POST['nombre']      ?? '');
    $formData['descripcion'] = trim($_POST['descripcion'] ?? '');

    // 2. Validaciones básicas
    if ($formData['nombre'] === '') {
        $errors[] = 'El nombre de la playlist es obligatorio.';
```

```
    }

    // 3. Manejo de imagen (opcional)
    $imagePath = null;

    if (!empty($_FILES['imagen']['name']) && $_FILES['imagen']['error'] !==
UPLOAD_ERR_NO_FILE) {
        if ($_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
            $uploadDir = __DIR__ . '/uploads/playlists/';
            if (!is_dir($uploadDir)) {
                mkdir($uploadDir, 0777, true);
            }

            $ext = strtolower(pathinfo($_FILES['imagen']['name'],
PATHINFO_EXTENSION));
            $allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];

            if (!in_array($ext, $allowed)) {
                $errors[] = 'Formato de imagen no permitido para la
playlist. Usa JPG, PNG, GIF o WEBP.';
            } else {
                $basename = 'pl_' . time() . '_' . mt_rand(1000, 9999) . '.'
. $ext;
                $destPath = $uploadDir . $basename;
                $webPath  = 'uploads/playlists/' . $basename; // ruta que se
guarda en la BD

                if (move_uploaded_file($_FILES['imagen']['tmp_name'],
$destPath)) {
                    $imagePath = $webPath;
                } else {
                    $errors[] = 'No se pudo subir la imagen de la
playlist.';
                }
            }
        } else {
            $errors[] = 'Ocurrió un error al subir la imagen de la
playlist.';
        }
    }

    // 4. Guardar si no hay errores
    if (empty($errors)) {
        $data = [
```

```
            'user_id'     => null,  // en este proyecto no manejamos
usuarios reales
            'name'        => $formData['nombre'],
            'description' => $formData['descripcion'] !== '' ?
$formData['descripcion'] : null,
            'image_path'  => $imagePath,
        ];

        if (createPlaylist($pdo, $data)) {
            header('Location: playlist.php?msg=creada');
            exit;
        } else {
            $errors[] = 'Ocurrió un error al crear la playlist en la base de
datos.';
        }
    }
}

// Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/crearPlaylistView.php';
require __DIR__ . '/views/partials/footer.php';
```

editar-playlist.php

```
<?php
// editar-playlist.php → Controlador para editar una playlist

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/PlaylistModel.php';

$pageTitle  = 'Editar Playlist - App Musical';
$activePage = 'playlist';

$errors   = [];
$formData = [
    'nombre'      => '',
    'descripcion' => '',
];

// 1. Validar ID
```

```php
if (empty($_GET['id']) || !ctype_digit($_GET['id'])) {
    header('Location: playlist.php');
    exit;
}

$id = (int) $_GET['id'];

// 2. Obtener playlist desde la BD
$playlist = getPlaylistById($pdo, $id);

if (!$playlist) {
    die("La playlist con ID $id no existe.");
}

// Imagen actual (puede ser null)
$currentImagePath = $playlist['image_path'] ?? null;

// 3. Obtener canciones asociadas actualmente a la playlist
$playlistSongIds = getPlaylistSongIds($pdo, $id);

// 4. Obtener todas las canciones disponibles para mostrarlas como opciones
$allSongs = [];
$stmt = $pdo->query("SELECT id, title, artist FROM songs ORDER BY title
ASC");
$allSongs = $stmt->fetchAll();

// 5. Rellenar formulario inicial con los datos de la playlist
$formData['nombre']      = $playlist['name']        ?? '';
$formData['descripcion'] = $playlist['description'] ?? '';

// 6. Procesar POST (si el usuario envía el formulario)
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Datos del formulario
    $formData['nombre']      = trim($_POST['nombre']      ?? '');
    $formData['descripcion'] = trim($_POST['descripcion'] ?? '');

    // Canciones seleccionadas (checkboxes)
    $selectedSongs = $_POST['songs'] ?? [];
    if (!is_array($selectedSongs)) {
        $selectedSongs = [];
    }

    // Validaciones básicas
```

```
    if ($formData['nombre'] === '') {
        $errors[] = 'El nombre de la playlist es obligatorio.';
    }

    // Manejo de imagen (opcional)
    // Por defecto se mantiene la imagen actual
    $imagePath = $currentImagePath;

    if (!empty($_FILES['imagen']['name']) && $_FILES['imagen']['error'] !==
UPLOAD_ERR_NO_FILE) {
        if ($_FILES['imagen']['error'] === UPLOAD_ERR_OK) {
            $uploadDir = __DIR__ . '/uploads/playlists/';
            if (!is_dir($uploadDir)) {
                mkdir($uploadDir, 0777, true);
            }

            $ext = strtolower(pathinfo($_FILES['imagen']['name'],
PATHINFO_EXTENSION));
            $allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];

            if (!in_array($ext, $allowed)) {
                $errors[] = 'Formato de imagen no permitido para la
playlist. Usa JPG, PNG, GIF o WEBP.';
            } else {
                $basename = 'pl_' . time() . '_' . mt_rand(1000, 9999) . '.'
. $ext;
                $destPath = $uploadDir . $basename;
                $webPath  = 'uploads/playlists/' . $basename;

                if (move_uploaded_file($_FILES['imagen']['tmp_name'],
$destPath)) {
                    $imagePath = $webPath;
                } else {
                    $errors[] = 'No se pudo subir la nueva imagen de la
playlist.';
                }
            }
        } else {
            $errors[] = 'Ocurrió un error al subir la imagen de la
playlist.';
        }
    }

    // Si no hay errores, actualizar en la BD
```

```
    if (empty($errors)) {
        $data = [
            'name'        => $formData['nombre'],
            'description' => $formData['descripcion'] !== '' ?
$formData['descripcion'] : null,
            'image_path'  => $imagePath,
        ];

        if (updatePlaylist($pdo, $id, $data)) {
            // Actualizar canciones asociadas
            syncPlaylistSongs($pdo, $id, $selectedSongs);

            header('Location: playlist.php?msg=editada');
            exit;
        } else {
            $errors[] = 'Ocurrió un error al actualizar la playlist en la
base de datos.';
        }
    }

    // Si hubo errores, mantenemos la selección de canciones del POST
    $playlistSongIds = array_map('intval', $selectedSongs);
}

// 7. Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/editarPlaylistView.php';
require __DIR__ . '/views/partials/footer.php';
```

Ayuda.php

```
<?php
// ayuda.php → Página de ayuda

require __DIR__ . '/config/db.php';   // <- IMPORTANTE para que el header
tenga $pdo

$pageTitle  = 'Ayuda - App Musical';
```

```php
$activePage = 'ayuda';

// Cargar layout MVC
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/ayudaView.php';   // tu contenido de ayuda
require __DIR__ . '/views/partials/footer.php';
```

Perfil.php

```php
<?php
// perfil.php → Perfil de usuario dummy con avatar

require __DIR__ . '/config/db.php';
require __DIR__ . '/models/UserModel.php';

$pageTitle  = 'Mi Perfil - App Musical';
$activePage = 'perfil';

$errors  = [];
$message = '';

// 1. Obtener perfil (id=1)
$userProfile = getUserProfile($pdo);
$currentAvatar = $userProfile['avatar_path'] ?? null;

// 2. Procesar subida de avatar si viene POST
if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (!empty($_FILES['avatar']['name']) && $_FILES['avatar']['error'] !==
UPLOAD_ERR_NO_FILE) {
        if ($_FILES['avatar']['error'] === UPLOAD_ERR_OK) {
            $uploadDir = __DIR__ . '/uploads/user/';
            if (!is_dir($uploadDir)) {
                mkdir($uploadDir, 0777, true);
            }

            $ext = strtolower(pathinfo($_FILES['avatar']['name'],
PATHINFO_EXTENSION));
            $allowed = ['jpg', 'jpeg', 'png', 'gif', 'webp'];

            if (!in_array($ext, $allowed)) {
```

```
                $errors[] = 'Formato de imagen no permitido. Usa JPG, PNG,
GIF o WEBP.';
            } else {
                $basename = 'avatar_' . time() . '_' . mt_rand(1000, 9999) .
'.' . $ext;
                $destPath = $uploadDir . $basename;
                $webPath  = 'uploads/user/' . $basename;

                if (move_uploaded_file($_FILES['avatar']['tmp_name'],
$destPath)) {
                    if (updateUserAvatar($pdo, $webPath)) {
                        $currentAvatar = $webPath;
                        $message = 'Avatar actualizado correctamente.';
                    } else {
                        $errors[] = 'No se pudo guardar el avatar en la base
de datos.';
                    }
                } else {
                    $errors[] = 'No se pudo subir el archivo de avatar.';
                }
            }
        } else {
            $errors[] = 'Ocurrió un error al subir el avatar.';
        }
    } else {
        $errors[] = 'No seleccionaste ninguna imagen para el avatar.';
    }
}

// 3. Cargar vistas
require __DIR__ . '/views/partials/header.php';
require __DIR__ . '/views/perfilView.php';
require __DIR__ . '/views/partials/footer.php';
```

SongModel.php

```
<?php
// models/SongModel.php

/**
 * Devuelve las canciones más recientes para el inicio.
 */
```

```php
function getRecentSongs(PDO $pdo, int $limit = 3): array
{
    $sql = "SELECT id, title, artist, genre, year, image_path
            FROM songs
            ORDER BY created_at DESC
            LIMIT :limit";

    $stmt = $pdo->prepare($sql);
    $stmt->bindValue(':limit', $limit, PDO::PARAM_INT);
    $stmt->execute();

    return $stmt->fetchAll();
}

/**
 * Lista canciones con filtros opcionales (buscar, género, artista, año).
 *
 * $filters = [
 *   'search' => 'texto a buscar en título o artista',
 *   'genre'  => 'Rock' | 'Pop' | 'Todos' | '',
 *   'artist' => 'nombre artista' | 'Todos' | '',
 *   'year'   => '2024' | '2023' | 'Todos' | ''
 * ]
 */
function getSongs(PDO $pdo, array $filters = []): array
{
    $where  = [];
    $params = [];

    $search = trim($filters['search'] ?? '');
    $genre  = trim($filters['genre']  ?? '');
    $artist = trim($filters['artist'] ?? '');
    $year   = trim($filters['year']   ?? '');

    if ($search !== '') {
        $where[] = "(title LIKE :search OR artist LIKE :search)";
        $params[':search'] = '%' . $search . '%';
    }

    if ($genre !== '' && $genre !== 'Todos') {
        $where[] = "genre = :genre";
        $params[':genre'] = $genre;
    }
```

```php
    if ($artist !== '' && $artist !== 'Todos') {
        $where[] = "artist = :artist";
        $params[':artist'] = $artist;
    }

    if ($year !== '' && $year !== 'Todos') {
        $where[] = "year = :year";
        $params[':year'] = (int)$year;
    }

    $sql = "SELECT id, title, artist, genre, year, image_path
            FROM songs";

    if ($where) {
        $sql .= " WHERE " . implode(' AND ', $where);
    }

    $sql .= " ORDER BY id DESC";

    $stmt = $pdo->prepare($sql);
    $stmt->execute($params);

    return $stmt->fetchAll();
}

/**
 * Elimina una canción por ID.
 */
function deleteSong(PDO $pdo, int $id): bool
{
    $stmt = $pdo->prepare("DELETE FROM songs WHERE id = :id");
    return $stmt->execute([':id' => $id]);
}

/**
 * Crea una nueva canción en la base de datos.
 */
function createSong(PDO $pdo, array $data): bool
{
    $sql = "INSERT INTO songs (title, artist, genre, year, album, duration,
image_path)
            VALUES (:title, :artist, :genre, :year, :album, :duration,
:image_path)";
```

```php
    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
        ':title'      => $data['title'],
        ':artist'     => $data['artist'],
        ':genre'      => $data['genre']    ?? null,
        ':year'       => $data['year']     ?? null,
        ':album'      => $data['album']    ?? null,
        ':duration'   => $data['duration'] ?? null,
        ':image_path' => $data['image_path'] ?? null,
    ]);
}

/**
 * Obtiene una canción por ID.
 */
function getSongById(PDO $pdo, int $id): ?array
{
    $stmt = $pdo->prepare("SELECT * FROM songs WHERE id = :id");
    $stmt->execute([':id' => $id]);
    $song = $stmt->fetch();

    return $song ?: null;
}

/**
 * Actualiza una canción existente.
 */
function updateSong(PDO $pdo, int $id, array $data): bool
{
    $sql = "UPDATE songs
            SET title      = :title,
                artist     = :artist,
                genre      = :genre,
                year       = :year,
                album      = :album,
                duration   = :duration,
                image_path = :image_path
            WHERE id = :id";

    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
```

```
        ':title'      => $data['title'],
        ':artist'     => $data['artist'],
        ':genre'      => $data['genre']    ?? null,
        ':year'       => $data['year']     ?? null,
        ':album'      => $data['album']    ?? null,
        ':duration'   => $data['duration'] ?? null,
        ':image_path' => $data['image_path'] ?? null,
        ':id'         => $id,
    ]);
}

/**
 * Devuelve lista de artistas distintos (para el combo de filtros).
 */
function getDistinctArtists(PDO $pdo): array
{
    $stmt = $pdo->query("SELECT DISTINCT artist FROM songs WHERE artist IS
NOT NULL AND artist <> '' ORDER BY artist ASC");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

/**
 * Devuelve lista de años distintos (para el combo de filtros).
 */
function getDistinctYears(PDO $pdo): array
{
    $stmt = $pdo->query("SELECT DISTINCT year FROM songs WHERE year IS NOT
NULL ORDER BY year DESC");
    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}
```

PlaylistModel.php

```
<?php
// models/PlaylistModel.php

/**
 * Obtiene todas las playlists con el conteo de canciones.
 */
function getPlaylists(PDO $pdo): array
```

```php
{
    $sql = "
        SELECT
            p.id,
            p.name,
            p.description,
            p.image_path,
            p.created_at,
            COUNT(ps.song_id) AS song_count
        FROM playlists p
        LEFT JOIN playlist_song ps ON p.id = ps.playlist_id
        GROUP BY p.id, p.name, p.description, p.image_path, p.created_at
        ORDER BY p.id DESC
    ";

    $stmt = $pdo->query($sql);
    return $stmt->fetchAll();
}

/**
 * Elimina una playlist por ID.
 * (Las relaciones en playlist_song se borran por la FK ON DELETE CASCADE)
 */
function deletePlaylist(PDO $pdo, int $id): bool
{
    $stmt = $pdo->prepare("DELETE FROM playlists WHERE id = :id");
    return $stmt->execute([':id' => $id]);
}

/**
 * Crea una nueva playlist.
 *
 * $data = [
 *   'user_id'     => int|null,
 *   'name'        => string,
 *   'description' => string|null,
 *   'image_path'  => string|null
 * ]
 */
function createPlaylist(PDO $pdo, array $data): bool
{
    $sql = "INSERT INTO playlists (user_id, name, description, image_path)
            VALUES (:user_id, :name, :description, :image_path)";
```

```php
    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
        ':user_id'     => $data['user_id']    ?? null,
        ':name'        => $data['name'],
        ':description' => $data['description'] ?? null,
        ':image_path'  => $data['image_path'] ?? null,
    ]);
}

/**
 * Obtiene una playlist por ID.
 */
function getPlaylistById(PDO $pdo, int $id): ?array
{
    $stmt = $pdo->prepare("SELECT * FROM playlists WHERE id = :id");
    $stmt->execute([':id' => $id]);
    $row = $stmt->fetch();

    return $row ?: null;
}

/**
 * Actualiza una playlist existente.
 *
 * $data = [
 *   'name'        => string,
 *   'description' => string|null,
 *   'image_path'  => string|null
 * ]
 */
function updatePlaylist(PDO $pdo, int $id, array $data): bool
{
    $sql = "UPDATE playlists
            SET name        = :name,
                description = :description,
                image_path  = :image_path
            WHERE id = :id";

    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
        ':name'        => $data['name'],
        ':description' => $data['description'] ?? null,
```

```
        ':image_path'   => $data['image_path'] ?? null,
        ':id'           => $id,
    ]);
}

/**
 * Devuelve los IDs de canciones asociadas a una playlist (para checkboxes).
 */
function getPlaylistSongIds(PDO $pdo, int $playlistId): array
{
    $stmt = $pdo->prepare("SELECT song_id FROM playlist_song WHERE
playlist_id = :id");
    $stmt->execute([':id' => $playlistId]);

    return $stmt->fetchAll(PDO::FETCH_COLUMN);
}

/**
 * Sincroniza las canciones de una playlist:
 * borra las actuales y agrega las nuevas (si se usan checkboxes).
 *
 * @param int   $playlistId
 * @param int[] $songIds
 */
function syncPlaylistSongs(PDO $pdo, int $playlistId, array $songIds): bool
{
    // Borrar relaciones actuales
    $stmt = $pdo->prepare("DELETE FROM playlist_song WHERE playlist_id =
:id");
    $stmt->execute([':id' => $playlistId]);

    // Insertar nuevas
    if (empty($songIds)) {
        return true;
    }

    $stmt = $pdo->prepare("INSERT INTO playlist_song (playlist_id, song_id)
VALUES (:playlist_id, :song_id)");

    foreach ($songIds as $songId) {
        if (!ctype_digit((string)$songId)) {
            continue;
        }
        $stmt->execute([
```

```
            ':playlist_id' => $playlistId,
            ':song_id'     => (int)$songId,
        ]);
    }

    return true;
}
```

UserModel.php

```
<?php
// models/UserModel.php

/**
 * Devuelve el perfil del usuario dummy (id 1).
 * Si no existe, intenta crearlo vacío.
 */
function getUserProfile(PDO $pdo): array
{
    $stmt = $pdo->prepare("SELECT * FROM user_profile WHERE id = 1");
    $stmt->execute();
    $row = $stmt->fetch();

    if (!$row) {
        $pdo->exec("INSERT INTO user_profile (id, avatar_path) VALUES (1,
NULL)");
        $stmt = $pdo->prepare("SELECT * FROM user_profile WHERE id = 1");
        $stmt->execute();
        $row = $stmt->fetch();
    }

    return $row ?: ['id' => 1, 'avatar_path' => null];
}

/**
 * Actualiza solo el avatar del usuario dummy.
 */
function updateUserAvatar(PDO $pdo, string $avatarPath): bool
{
```

```
    $stmt = $pdo->prepare("UPDATE user_profile SET avatar_path = :avatar
WHERE id = 1");
    return $stmt->execute([':avatar' => $avatarPath]);
}
```

Db.php

```
<?php
// config/db.php

// Detectamos si estamos en InfinityFree (servidor) o en XAMPP (localhost)
$isProduction = (isset($_SERVER['HTTP_HOST']) && $_SERVER['HTTP_HOST'] !==
'localhost');

// Configuración dependiendo del entorno
if ($isProduction) {
    // === DATOS DE INFINITYFREE ===
    $host = 'sql303.infinityfree.com';
    $db   = 'if0_40584663_app_musical';
    $user = 'if0_40584663';
    $pass = 'PROYECTO12345';
} else {
    // === DATOS DE XAMPP (LOCALHOST) ===
    $host = 'localhost';
    $db   = 'app_musical';
    $user = 'root';
    $pass = '';
}

// Construcción del DSN
$dsn = "mysql:host=$host;dbname=$db;charset=utf8mb4";

// Opciones de PDO
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];

// Intento de conexión
try {
    $pdo = new PDO($dsn, $user, $pass, $options);
```

```php
} catch (PDOException $e) {
    die('Error de conexión a la base de datos: ' . $e->getMessage());
}
```

Tabla de Aportes

Fase 4: BackEnd

Parte/Actividad	Fecha de Realización	Miembro del Equipo Responsable
Configuración del archivo db.php y prueba de conexión PDO a MySQL.	01/12/2025	Diego Corrales
Implementación del archivo SongModel.php con CRUD de canciones.	01/12/2025	Daniel Pérez
Implementación del archivo PlaylistModel.php con CRUD y conteo.	02/12/2025	Juan Copri
Implementación del archivo UserModel.php para gestión de usuarios.	02/12/2025	Josué Pino
Integración de header.php y footer.php en todas las vistas.	03/12/2025	Diego Corrales
Lógica dinámica en index.php e indexView.php.	03/12/2025	Daniel Pérez
Listado dinámico de canciones (canciones.php y cancionesView.php).	04/12/2025	Juan Copri
Creación de playlists con validaciones.	04/12/2025	Josué Pino
Funcionalidad para agregar canciones a playlists.	05/12/2025	Diego Corrales
Edición de canciones con precarga desde BD.	05/12/2025	Daniel Pérez
Edición de playlists con actualización de datos.	06/12/2025	Juan Copri
Página de perfil con estadísticas y actualización.	06/12/2025	Josué Pino