Université Paris Nanterre Université Paris

Nanterre

L2 Informatique

Rapport de Projet Informatique

SmartMix

Générateur de Playlist Web

Membres du groupe :

Moutaouakkil Mohamed Reda - 44004006
Garbaa Hamza - 44013028

Lien vers le dépôt GitHub :
`https://github.com/hamza-garbaa76/SmartMix.git`

Semestre 3 - Année 2025/2026

Table des matières

1 Introduction

L'écoute musicale en ligne repose souvent sur des algorithmes complexes et opaques. Avec notre projet **SmartMix**, nous avons voulu créer une solution Web transparente et rapide.

SmartMix est une application web développée en PHP 8 qui permet de générer des playlists sur-mesure en temps réel. Contrairement aux solutions statiques, notre application interroge en direct les bases de données d'**iTunes** et d'**Internet Archive** pour offrir une variété musicale instantanée basée sur les choix de l'utilisateur (style, époque, durée).

2 Environnement de travail

Le projet a été réalisé en binôme, en alternant développement local et collaboration à distance.

- **Langage Backend :** PHP 8.x (Gestion de la logique et des API).
- **Langage Frontend :** HTML5, CSS3 (Interface responsive).
- **Outils :** Visual Studio Code, Git/GitHub.
- **Serveur :** WAMP / XAMPP pour l'exécution locale.

3 Description du projet et objectifs

L'objectif est de concevoir un "agrégateur musical" capable de construire une playlist cohérente à partir de sources multiples. Les fonctionnalités principales sont :

1. **Multi-critères :** Sélection multiple de styles (Rock, Jazz...) et d'époques (décennies).
2. **Génération Live :** Utilisation des API iTunes (pour la rapidité) et Internet Archive (pour la rareté).
3. **Algorithme de durée :** Remplissage intelligent pour atteindre la durée cible (ex : 60 min) avec une tolérance de ± 1 minute.
4. **Export :** Possibilité de télécharger la playlist générée au format CSV.

4 Bibliothèques et APIs utilisées

Nous avons choisi de ne pas utiliser de base de données locale, mais de tout récupérer dynamiquement.

- **iTunes Search API :** Fournit des extraits de 30 secondes et des métadonnées très propres. Très rapide.
- **Internet Archive API :** Fournit des titres souvent complets et libres de droits.
- **cURL (PHP) :** Bibliothèque utilisée pour effectuer les requêtes HTTP vers ces APIs.

5 Travail réalisé

5.1 Architecture et Code

Le cœur de l'application repose sur l'interaction avec plusieurs services et une gestion de données locale efficace.

5.1.1 Récupération API (iTunes)

Voici le code gérant l'interaction avec l'API iTunes :

```php
<?php
declare(strict_types=1);

function itunes_http_get_json(string $url, int $timeout = 20): ?array {
  $ctx = stream_context_create([
    'http' => [
      'method' => 'GET',
      'timeout' => $timeout,
      'header' => "User-Agent: PlaylistBuilder/1.0\r\nAccept: application/json\r\n",
    ],
  ]);

  $raw = @file_get_contents($url, false, $ctx);
  if ($raw === false) return null;

  $data = json_decode($raw, true);
  return is_array($data) ? $data : null;
}
// ... (Fonctions de filtrage omises pour brievet )
```

Listing 1 – Gestion de l'API iTunes (itunes.php)

Analyse : Ce script gère l'interaction avec l'API publique d'iTunes via des requêtes HTTP GET. Il filtre les résultats pour ne conserver que les morceaux correspondant à la décennie demandée et convertit les durées (ms) en secondes pour l'algorithme.

5.1.2 Complexité et Cache (Internet Archive)

Contrairement à iTunes, l'API d'Internet Archive est plus lente. Nous avons implémenté un cache local :

```php
<?php
declare(strict_types=1);

function ia_get_metadata_cached(string $identifier, string $cacheDir, int $ttlSec): ?
    array {
  ia_ensure_dir($cacheDir);
  $safe = preg_replace('~[^a-zA-Z0-9._-]+~', '_', $identifier);
  $path = rtrim($cacheDir, '/').'/'.$safe.'.json';

  // Verification du cache (Time To Live)
  if (is_file($path)) {
    $age = time() - (int)@filemtime($path);
    if ($age >= 0 && $age < $ttlSec) {
      $raw = @file_get_contents($path);
      if ($raw !== false) {
        $data = json_decode($raw, true);
        if (is_array($data)) return $data;
      }
    }
  }

  // Si pas en cache, on appelle l'API
  $data = ia_get_metadata($identifier);
  if (is_array($data)) {
    @file_put_contents($path, json_encode($data));
```

```
  }
  return $data;
}

function ia_extract_mp3_tracks_from_metadata(array $meta, string $styleTerm, int $yStart,
    int $yEnd): array {
  // Parsing complexe pour trouver les fichiers .mp3 dans les metadata
  $identifier = $meta['metadata']['identifier'] ?? null;
  // ... (Logique de filtrage des annees et formats)
  foreach ($files as $f) {
    if (stripos($format, 'mp3') === false) continue;
    // ...
  }
  return $out;
}
```

Listing 2 – Gestion avancée Internet Archive (archive_api.php)

Analyse : Ce module gère la complexité de l'Internet Archive. La fonction `ia_get_metadata_cached` stocke les métadonnées JSON localement pour 7 jours, évitant les lenteurs.

5.1.3 Logique Algorithmique (SmartMix Core)

Le "cerveau" du site assemble la playlist :

```
<?php
function smx_build_playlist(array $candidates, int $targetSec, int $seed): array {
  $tolerance = 60;  // allow +60s
  $maxTracks = 60;
  smx_seeded_shuffle($candidates, $seed);

  $playlist = [];
  $sum = 0;

  foreach ($candidates as $t) {
    $d = (int)($t['duration_sec'] ?? 0);
    if ($sum + $d <= $targetSec + $tolerance) {
      $playlist[] = $t;
      $sum += $d;
    }
    // ...
  }
  // (Optimisation finale pour echanger le dernier titre et gagner en precision)
  return ['tracks' => array_values($best), 'total_sec' => (int)$bestSum];
}
```

Listing 3 – Algorithme de remplissage (smartmix_core.php)

Analyse : L'algorithme principal utilise une approche gloutonne pour remplir la playlist jusqu'à la durée cible, avec une étape d'optimisation finale pour minimiser l'écart de temps.

5.1.4 Persistance des Données

Sauvegarde via fichiers JSON sécurisés par token :

```
<?php
declare(strict_types=1);

function pl_save(string $dir, array $payload): string {
  pl_ensure_dir($dir);
  // Generation d'un token unique aleatoire
  $token = bin2hex(random_bytes(16));
  $path = rtrim($dir, '/') . '/' . $token . '.json';

  // Sauvegarde des donnees en JSON
  file_put_contents($path, json_encode($payload, JSON_UNESCAPED_UNICODE));
  return $token;
}
```

```php

function pl_load(string $dir, string $token): ?array {
  // Verification de securite du format token
  if (!preg_match('~^[a-f0-9]{32}$~', $token)) return null;

  $path = rtrim($dir, '/') . '/' . $token . '.json';
  if (!is_file($path)) return null;

  $raw = file_get_contents($path);
  $data = json_decode($raw, true);
  return is_array($data) ? $data : null;
}
```

Listing 4 – Système de sauvegarde fichier (storage.php)

Analyse : Chaque playlist est sauvegardée dans un fichier JSON avec un nom aléatoire (token). Cela permet le partage via URL sans base de données SQL.

5.2 Collaboration avec l'IA (Section Obligatoire)

Nous avons utilisé l'IA (ChatGPT/Gemini) comme assistant technique :
— **Aide API :** L'IA nous a aidés à comprendre la structure JSON d'iTunes.
— **Débogage :** Résolution de problèmes cURL et regex.

6 Difficultés rencontrées

— **Hétérogénéité des données :** Il a fallu uniformiser les durées (ms vs "mm :ss") entre les deux APIs.
— **Temps de réponse :** La génération peut être lente à cause des appels externes.
— **Pertinence :** Le filtrage des résultats d'Internet Archive a demandé beaucoup d'ajustements.

7 Bilan

7.1 Conclusion

Ce projet nous a permis de manipuler des APIs REST, de traiter du JSON en PHP et de gérer les contraintes du web. L'export CSV rend l'outil fonctionnel.

7.2 Perspectives

Améliorations possibles : système de cache plus agressif et lecture enchaînée (autoplay).

8 Webographie

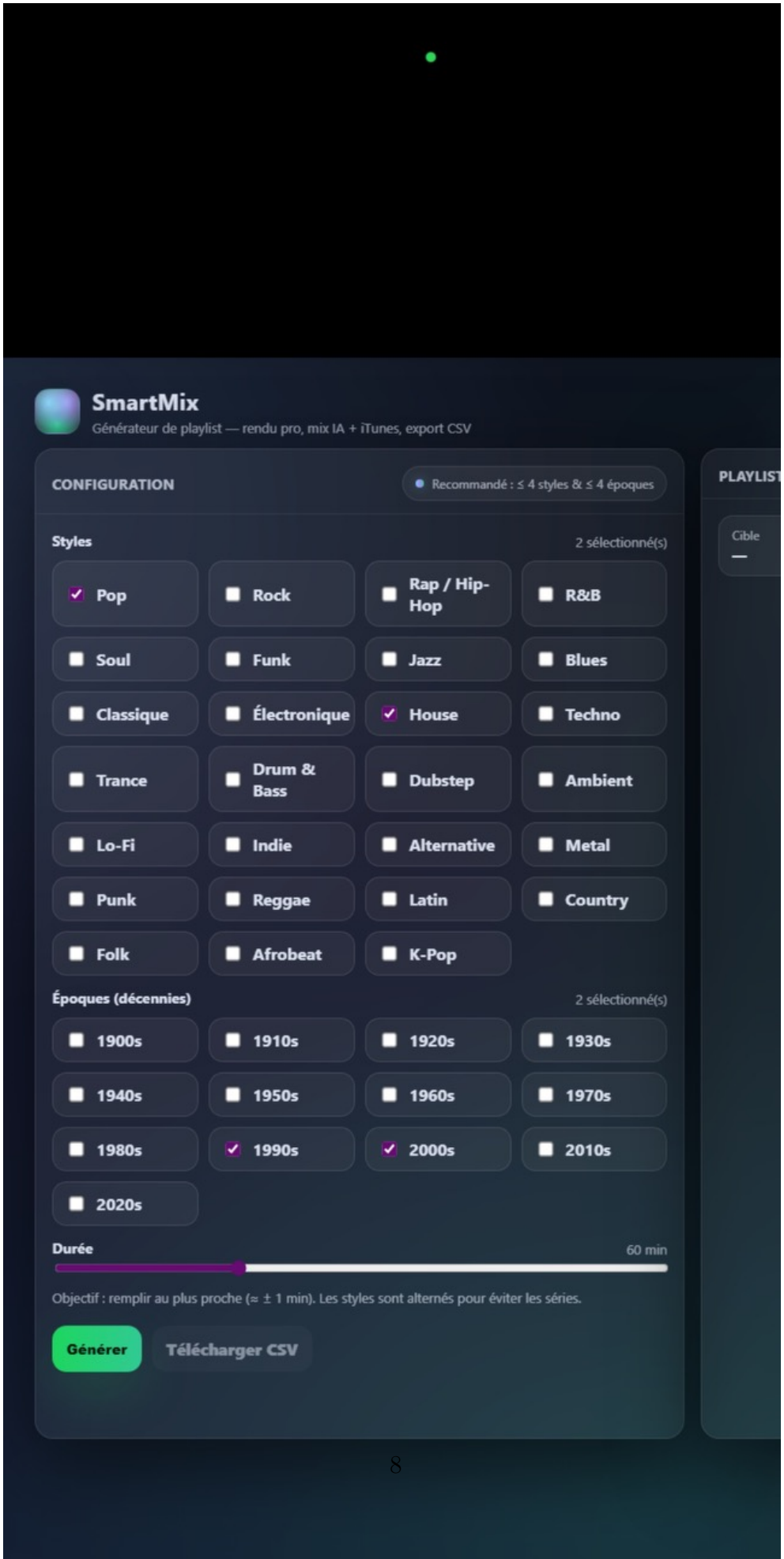
— Documentation PHP : `https://www.php.net/`
— iTunes Search API Internet Archive API.

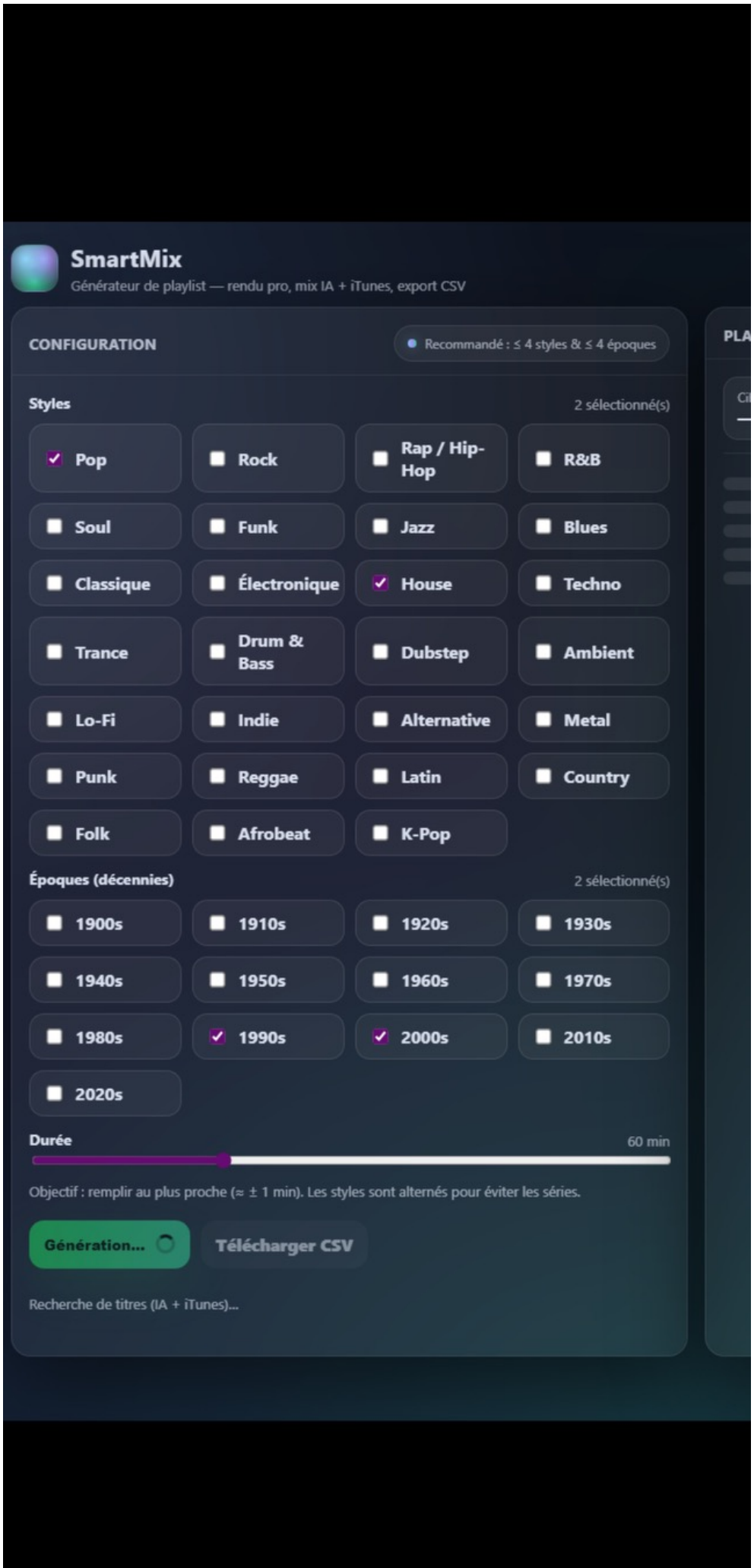
9 Annexes

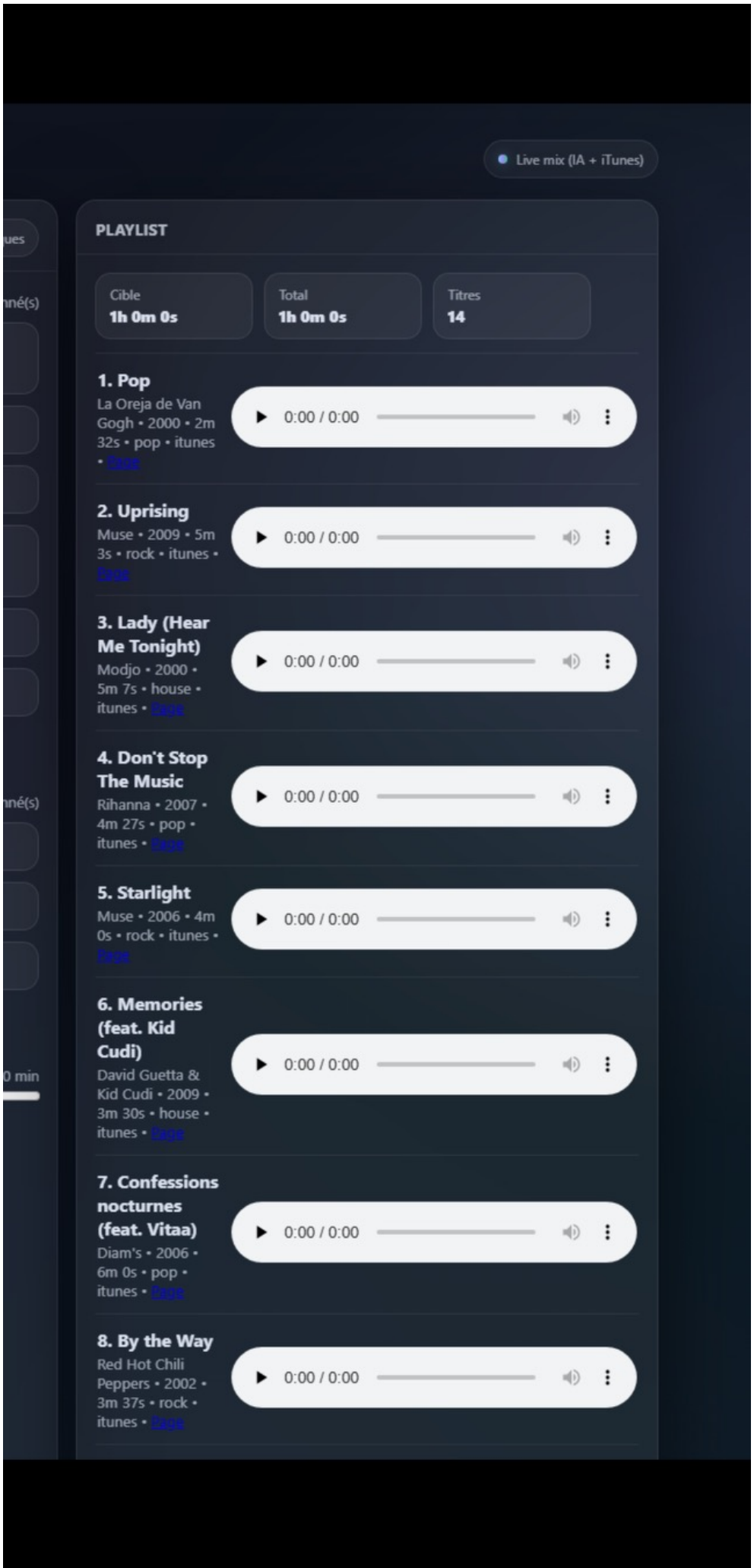
.1 Annexe A : Cahier des charges technique

Entrées : Styles, Époque, Durée. **Sorties :** Playlist HTML + CSV.

.2 Annexe B : Aperçu de l'interface




SmartMix
Générateur de playlist — rendu pro, mix IA + iTunes, export CSV
CONFIGURATION
Recommandé : ≤ 4 styles & ≤ 4 époques
Styles
2 sélectionné(s)
Pop
Rock
Rap / Hip-Hop
R&B
Soul
Funk
Jazz
Blues
Classique
Électronique
House
Techno
Trance
Drum & Bass
Dubstep
Ambient
Lo-Fi
Indie
Alternative
Metal
Punk
Reggae
Latin
Country
Folk
Afrobeat
K-Pop
Époques (décennies)
2 sélectionné(s)
1900s
1910s
1920s
1930s
1940s
1950s
1960s
1970s
1980s
1990s
2000s
2010s
2020s
Durée
60 min
Objectif : remplir au plus proche (≈ ± 1 min). Les styles sont alternés pour éviter les séries.
Génération...
Télécharger CSV
Recherche de titres (IA + iTunes)...


Live mix (IA + iTunes)
PLAYLIST
Cible
1h 0m 0s
Total
1h 0m 0s
Titres
14
1. Pop
La Oreja de Van Gogh • 2000 • 2m 32s • pop • itunes • Page
0:00 / 0:00
2. Uprising
Muse • 2009 • 5m 3s • rock • itunes • Page
0:00 / 0:00
3. Lady (Hear Me Tonight)
Modjo • 2000 • 5m 7s • house • itunes • Page
0:00 / 0:00
4. Don't Stop The Music
Rihanna • 2007 • 4m 27s • pop • itunes • Page
0:00 / 0:00
5. Starlight
Muse • 2006 • 4m 0s • rock • itunes • Page
0:00 / 0:00
6. Memories (feat. Kid Cudi)
David Guetta & Kid Cudi • 2009 • 3m 30s • house • itunes • Page
0:00 / 0:00
7. Confessions nocturnes (feat. Vitaa)
Diam's • 2006 • 6m 0s • pop • itunes • Page
0:00 / 0:00
8. By the Way
Red Hot Chili Peppers • 2002 • 3m 37s • rock • itunes • Page
0:00 / 0:00

.3 Annexe C : Structure CSV

Colonnes : Titre, Artiste, Durée, URL.