

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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Environnement de travail</b>	<b>2</b>
<b>3</b>	<b>Description du projet et objectifs</b>	<b>2</b>
<b>4</b>	<b>Bibliothèques et APIs utilisées</b>	<b>2</b>
<b>5</b>	<b>Travail réalisé</b>	<b>3</b>
5.1	Architecture et Code . . . . .	3
5.1.1	Récupération API (iTunes) . . . . .	3
5.1.2	Complexité et Cache (Internet Archive) . . . . .	3
5.1.3	Logique Algorithmique (SmartMix Core) . . . . .	4
5.1.4	Persistance des Données . . . . .	4
5.2	Collaboration avec l'IA . . . . .	5
5.2.1	Présentaton de notre IA . . . . .	5
5.2.2	Utilisation de notre IA . . . . .	5
5.3	Tâches non réalisées . . . . .	5
5.4	Répartition des tâches . . . . .	5
<b>6</b>	<b>Difficultés rencontrées</b>	<b>6</b>
<b>7</b>	<b>Bilan</b>	<b>6</b>
7.1	Conclusion . . . . .	6
7.2	Perspectives . . . . .	6
<b>8</b>	<b>Webographie</b>	<b>6</b>
<b>9</b>	<b>Annexes</b>	<b>6</b>
.1	Annexe A : Cahier des charges technique . . . . .	6
.2	Annexe B : Aperçu de l'interface . . . . .	8
.3	Annexe C : Structure CSV . . . . .	11

# 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  $\pm 1$  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 :

```
1 <?php
2 declare(strict_types=1);
3
4 function itunes_http_get_json(string $url, int $timeout = 20): ?array {
5     $ctx = stream_context_create([
6         'http' => [
7             'method' => 'GET',
8             'timeout' => $timeout,
9             'header' => "User-Agent: PlaylistBuilder/1.0\r\nAccept: application/json\r\n",
10        ],
11    );
12
13     $raw = @file_get_contents($url, false, $ctx);
14     if ($raw === false) return null;
15
16     $data = json_decode($raw, true);
17     return is_array($data) ? $data : null;
18 }
19 // ... (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 :

```
1 <?php
2 declare(strict_types=1);
3
4 function ia_get_metadata_cached(string $identifier, string $cacheDir, int $ttlSec): ?
5     array {
6     ia.ensure_dir($cacheDir);
7     $safe = preg_replace('^[^a-zA-Z0-9._-]+$', '_', $identifier);
8     $path = rtrim($cacheDir, '/') . '/' . $safe . '.json';
9
10    // Verification du cache (Time To Live)
11    if (is_file($path)) {
12        $age = time() - (int)@filemtime($path);
13        if ($age >= 0 && $age < $ttlSec) {
14            $raw = @file_get_contents($path);
15            if ($raw !== false) {
16                $data = json_decode($raw, true);
17                if (is_array($data)) return $data;
18            }
19        }
20
21    // Si pas en cache, on appelle l'API
22    $data = ia_get_metadata($identifier);
23    if (is_array($data)) {
24        @file_put_contents($path, json_encode($data));
```

```

25     }
26     return $data;
27 }
28
29 function ia_extract_mp3_tracks_from_metadata(array $meta, string $styleTerm, int $yStart,
30     int $yEnd): array {
31     // Parsing complexe pour trouver les fichiers .mp3 dans les metadata
32     $identifier = $meta['metadata']['identifier'] ?? null;
33     // ... (Logique de filtrage des années et formats)
34     foreach ($files as $f) {
35         if (stripos($format, 'mp3') === false) continue;
36         // ...
37     }
38 }

```

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 :

```

1 <?php
2 function smx_build_playlist(array $candidates, int $targetSec, int $seed): array {
3     $tolerance = 60; // allow +60s
4     $maxTracks = 60;
5     smx_seeded_shuffle($candidates, $seed);
6
7     $playlist = [];
8     $sum = 0;
9
10    foreach ($candidates as $t) {
11        $d = (int)($t['duration_sec'] ?? 0);
12        if ($sum + $d <= $targetSec + $tolerance) {
13            $playlist[] = $t;
14            $sum += $d;
15        }
16        // ...
17    }
18    // (Optimisation finale pour échanger le dernier titre et gagner en précision)
19    return ['tracks' => array_values($best), 'total_sec' => (int)$bestSum];
20 }

```

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 :

```

1 <?php
2 declare(strict_types=1);
3
4 function pl_save(string $dir, array $payload): string {
5     pl_ensure_dir($dir);
6     // Génération d'un token unique aléatoire
7     $token = bin2hex(random_bytes(16));
8     $path = rtrim($dir, '/') . '/' . $token . '.json';
9
10    // Sauvegarde des données en JSON
11    file_put_contents($path, json_encode($payload, JSON_UNESCAPED_UNICODE));
12    return $token;
13 }

```

```

14
15 function pl_load(string $dir, string $token): ?array {
16     // Vérification de sécurité du format token
17     if (!preg_match('~~[a-f0-9]{32}~~', $token)) return null;
18
19     $path = rtrim($dir, '/') . '/' . $token . '.json';
20     if (!is_file($path)) return null;
21
22     $raw = file_get_contents($path);
23     $data = json_decode($raw, true);
24     return is_array($data) ? $data : null;
25 }

```

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

### 5.2.1 Présentation de notre IA

Gemini est une famille de modèles d'intelligence artificielle développée par Google, reposant sur les modèles de langage de grande taille (Large Language Models). Son fonctionnement est basé sur des réseaux de neurones profonds capables d'analyser et de générer du langage naturel. L'une des principales caractéristiques de Gemini est son aspect multimodal : le modèle peut traiter simultanément différents types de données tels que le texte, les images ou le code. Lors de son entraînement, Gemini apprend à identifier des régularités statistiques à partir de grandes quantités de données, afin de prédire la suite la plus probable d'une séquence en fonction du contexte. Lorsqu'une requête est soumise, celle-ci est convertie en une représentation mathématique permettant au modèle d'en comprendre le sens et l'intention. La réponse est ensuite générée de manière progressive, mot par mot, en tenant compte du contexte global. Bien que performant, Gemini repose sur des calculs probabilistes et ne possède pas de compréhension consciente, ce qui implique la nécessité de garder un esprit critique quant aux réponses produites. Si tu veux, je peux encore réduire à un seul paragraphe ou l'adapter exactement au nombre de lignes demandé.

### 5.2.2 Utilisation de notre IA

Nous avons utilisé l'IA (Gemini) comme assistant technique :

- **Aide API :** L'IA nous a aidés à comprendre la structure JSON d'iTunes, à manipuler le codage du Latex et du site et aussi rédiger le Read.me.
- **Débogage :** Résolution de problèmes URL et regex.

## 5.3 Tâches non réalisées

Nous ne pouvons pas écouter les musiques directement sur le site en entier (que des extraits), la lecture n'est pas enchaînée. Si le style n'est pas trouvé à telle époque, pas d'alternative de playlist proposée.

## 5.4 Répartition des tâches

Hamza a eu l'idée du site et du projet, Reda a fait le choix de l'IA. Ils ont donc mis leurs idées en commun et collaboré pour créer le projet. Hamza et Reda, grâce à Gemini

ont codé le site ensemble et l'ont testé. Hamza s'est chargé des API, du JSON tandis que Reda plus du php et du "squelette". Ensuite, Reda a rédigé la majorité du rapport et Hamza s'est chargé du GitHub.

## 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. Ce projet permet de générer des playlists intelligentes et permet un gain de temps en quelques clics.

### 7.2 Perspectives

Améliorations possibles : système de cache plus agressif et lecture enchaînée (autoplay). Créer une application est possible et on peu élargir le choix des musiques avec une plus grande BD.

## 8 Webographie

- Documentation PHP : <https://www.php.net/>
- iTunes Search API Internet Archive API.
- Gemini

## 9 Annexes

### .1 Annexe A : Cahier des charges technique

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

- input utilisateur avec saisie des préférences
- Moteur algorithmique avec parcours de la BD et sélection
- Affichage des statistiques, musiques et bouton pour télécharger la playlist.



## .2 Annexe B : Aperçu de l'interface

The screenshot shows the SmartMix application interface. At the top, there's a dark header with a green circular icon and the text "SmartMix". Below it, a sub-header reads "Générateur de playlist — rendu pro, mix IA + iTunes, export CSV".

The main area is divided into two sections: "CONFIGURATION" on the left and "PLAYLIST" on the right.

**CONFIGURATION:**

- Styles:** A grid of 20 music styles. "Pop" and "House" are checked. Other styles include Rock, Rap / Hip-Hop, R&B, Soul, Funk, Jazz, Blues, Classique, Électronique, Techno, Trance, Drum & Bass, Dubstep, Ambient, Lo-Fi, Indie, Alternative, Metal, Punk, Reggae, Latin, Country, Folk, Afrobeat, and K-Pop. A note indicates "2 sélectionné(s)".
- Époques (décennies):** A grid of 12 time periods. "1990s" and "2000s" are checked. Other periods include 1900s, 1910s, 1920s, 1930s, 1940s, 1950s, 1960s, 1970s, 1980s, 2010s, and 2020s. A note indicates "2 sélectionné(s)".
- Durée:** A horizontal slider set to "60 min".

**PLAYLIST:** This section is currently empty, indicated by a minus sign.

At the bottom, there are two buttons: "Générer" (Generate) and "Télécharger CSV" (Download CSV).

A note at the bottom states: "Objectif : remplir au plus proche ( $\approx \pm 1$  min). Les styles sont alternés pour éviter les séries."

# 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 ( $\approx \pm 1$  min). Les styles sont alternés pour éviter les séries.

**Génération...**

**Télécharger CSV**

Recherche de titres (IA + iTunes)...

Live mix (IA + iTunes)

## PLAYLIST

Cible	Total	Titres
<b>1h 0m 0s</b>	<b>1h 0m 0s</b>	<b>14</b>

**1. Pop**  
La Oreja de Van Gogh • 2000 • 2m 32s • pop • itunes • [Page](#)

**2. Uprising**  
Muse • 2009 • 5m 3s • rock • itunes • [Page](#)

**3. Lady (Hear Me Tonight)**  
Modjo • 2000 • 5m 7s • house • itunes • [Page](#)

**4. Don't Stop The Music**  
Rihanna • 2007 • 4m 27s • pop • itunes • [Page](#)

**5. Starlight**  
Muse • 2006 • 4m 0s • rock • itunes • [Page](#)

**6. Memories (feat. Kid Cudi)**  
David Guetta & Kid Cudi • 2009 • 3m 30s • house • itunes • [Page](#)

**7. Confessions nocturnes (feat. Vitaa)**  
Diam's • 2006 • 6m 0s • pop • itunes • [Page](#)

**8. By the Way**  
Red Hot Chili Peppers • 2002 • 3m 37s • rock • itunes • [Page](#)

### **.3 Annexe C : Structure CSV**

Colonnes : Titre, Artiste, Durée, URL.