Mini-projet du module Transmission de Données Multimédia.

Option : Informatique, Réseaux et Multimédias

**Étude Comparative des Techniques de Compression et développement du nouveau format IRM**

|  |
| --- |
|  |

**Réalisé** **par :**

* Yassir Beloukid
* Oumaima Youti
* Achraf Chahin
* Maroua Jettioui

**Encadré par :**

* Pr.Abdellah Adib

Année universitaire : **2024 – 2025**

Remerciements

Nous tenons à exprimer notre sincère gratitude au Professeur Abdellah ADIB pour son encadrement précieux et ses conseils avisés tout au long de ce mini-projet sur la comparaison des formats de compression audio.

Nos remerciements s'adressent également à la Faculté des Sciences et Techniques de Mohammedia de l'Université Hassan II de Casablanca pour les ressources mises à notre disposition et l'environnement propice à l'apprentissage qu'elle nous offre.

Nous souhaitons aussi remercier nos camarades de la Licence IRM pour les échanges enrichissants qui ont contribué à la réussite de ce travail.

L'esprit d'équipe et la collaboration entre les quatre membres de notre groupe ont été des atouts majeurs dans la réalisation de ce projet. La complémentarité de nos compétences nous a permis de relever les défis techniques et de proposer une solution innovante avec le format IRM.

Enfin, nous tenons à exprimer notre reconnaissance envers toutes les personnes qui ont participé, directement ou indirectement, à l'aboutissement de ce projet, notamment pour leurs suggestions pertinentes et leur soutien constant

Table des matières

[2 Introduction 5](#_Toc196085817)

[3 Etude théorique : 8](#_Toc196085818)

[**3.1** **Principe de compression audio avec perte et sans perte :** 8](#_Toc196085819)

[**3.1.1** **Compression avec perte:** 8](#_Toc196085820)

[**3.1.2** **Compression sans perte :** 9](#_Toc196085821)

[**3.1.3** **Comparaison des Deux Types de Compression :** 10](#_Toc196085822)

[**3.2** **Description des formats OGG et ALAC :** 11](#_Toc196085823)

[**3.2.1** **Format OGG :** 11](#_Toc196085824)

[**3.2.2** **Format ALAC :** 18](#_Toc196085825)

[**3.2.3** **Comparaison générale :** 24](#_Toc196085826)

[4 Extension personnalisée « .irm »: 25](#_Toc196085827)

[**4.1** **Analyse et planification :** 25](#_Toc196085828)

[**4.1.1** **Introduction à l'extension IRM :** 25](#_Toc196085829)

[**4.1.2** **Contexte et objectifs du format IRM** 25](#_Toc196085830)

[**4.1.3** **Présentation de la problématique :** 25](#_Toc196085831)

[**4.1.4** **Étude de l'existant :** 26](#_Toc196085832)

[**4.1.5** **Solution proposée :** 26](#_Toc196085833)

[**4.1.6** **Cahier des charges** 26](#_Toc196085834)

[**4.2** **Réalisation de l’extension personnalisée « .irm »:** 28](#_Toc196085835)

[**4.2.1** **Introduction :** 28](#_Toc196085836)

[**4.2.2** **Format IRM : Architecture et caractéristiques :** 29](#_Toc196085837)

[**4.2.3** **Implémentation des algorithmes de compression** 30](#_Toc196085838)

[**4.3** **Conception de l’interface graphique :** 38](#_Toc196085839)

[**4.3.1** **Introduction :** 38](#_Toc196085840)

[**4.3.2** **Langage de modélisation UML :** 38](#_Toc196085841)

[**4.3.3** **Logiciel utilisé :** 39](#_Toc196085842)

[**4.3.4** **Identification des acteurs :** 39](#_Toc196085843)

[**4.3.5** **Diagramme de cas d’utilisation.** 40](#_Toc196085844)

[**4.3.6** **Diagramme de séquences.** 42](#_Toc196085845)

[**4.3.7** **Conclusion :** 43](#_Toc196085846)

[**4.4** **Outils Techniques :** 44](#_Toc196085847)

[**4.5** **Réalisation de l’interface graphique :** 46](#_Toc196085848)

[**4.5.1** **Introduction :** 46](#_Toc196085849)

[**4.5.2** **Nom et logo de l’interface :** 47](#_Toc196085850)

[**4.5.3** **Interface de connexion et inscription :** 48](#_Toc196085851)

[**4.5.4** **Interface de compression pour Chercheur, Ingénieur du son et Développeur :** 50](#_Toc196085852)

[**4.5.5** **Interface principale de compression pour l’utilisateur standard :** 54](#_Toc196085853)

[**4.5.6** **Système d'aide contextuelle :** 55](#_Toc196085854)

[**4.5.7** **Personnalisation de l'expérience visuelle :** 56](#_Toc196085855)

[**4.5.8** **Innovation : Plateforme web de distribution :** 60](#_Toc196085856)

[**4.6** **Résultats et évaluation** 64](#_Toc196085857)

[**4.6.1** **Performances du format IRM :** 64](#_Toc196085858)

[**4.6.2** **Analyse comparative avec les formats existants :** 68](#_Toc196085859)

[**4.6.3** **Contraintes et limitations rencontrées :** 72](#_Toc196085860)

[5 Conclusion générale : 74](#_Toc196085861)

[6 Bibliographie : 75](#_Toc196085862)

[7 Webographie : 76](#_Toc196085863)

# Introduction

La compression audio est une technique essentielle dans le domaine du traitement des données numériques, permettant de réduire la taille des fichiers audio pour le stockage et la transmission. Elle joue un rôle crucial dans diverses applications, telles que les services de streaming musical, les systèmes de télécommunication et le stockage de données, facilitant ainsi l'échange et la gestion des fichiers audio.

L'émergence d'Internet et l'augmentation exponentielle de la production et de la consommation de contenu audio ont rendu la compression audio indispensable. Au cours des années 1990, le format MP3 a révolutionné le partage de musique en ligne en introduisant une méthode de compression avec perte qui réduisait la taille des fichiers tout en maintenant une qualité sonore acceptable pour les utilisateurs. Cette évolution a ouvert la voie à une multitude de formats de compression, chacun avec ses propres avantages et inconvénients.

Il existe deux grandes catégories de méthodes de compression audio : la compression sans perte (lossless) et la compression avec perte (lossy). La compression sans perte permet de réduire la taille des fichiers audio sans aucune perte d'information, garantissant une qualité sonore parfaite. Les formats courants incluent FLAC, ALAC et WAV. En revanche, la compression avec perte réduit la taille des fichiers en supprimant certaines informations jugées moins perceptibles pour l'oreille humaine, ce qui entraîne une perte de qualité sonore. Les formats populaires incluent MP3, AAC et Ogg Vorbis, largement utilisés dans les services de streaming pour leur efficacité en termes de réduction de la taille des fichiers.

L'objectif principal de ce projet est d'étudier et de comparer les différentes techniques de compression audio, en se concentrant particulièrement sur les formats OGG et ALAC. Le projet se divise en plusieurs étapes théoriques et pratiques. Tout d'abord, nous rappelons les principes de base de la compression audio, en explicitant les étapes de mise en place pour les méthodes avec et sans perte, et en détaillant les algorithmes assignés. Ensuite, nous mettons en œuvre des fonctions en Python pour transformer et comparer des fichiers audios entre différents formats et pour décompresser ces fichiers. Enfin, nous proposons une nouvelle méthode de compression innovante, nommée IRM, qui utilise une combinaison des codages RLE, Huffman et LZW.

La compression audio est omniprésente dans notre quotidien. Les services de streaming musical utilisent des formats avec perte pour offrir une expérience utilisateur fluide tout en minimisant la bande passante requise. Les solutions de stockage numérique utilisent des formats sans perte pour garantir la qualité des enregistrements. Dans les télécommunications, la compression audio permet de transmettre des voix avec une bonne qualité et une faible latence.

Malgré les nombreux avantages offerts par les différentes techniques de compression, des défis subsistent, notamment en termes de qualité sonore et de complexité algorithmique. Les innovations récentes se concentrent sur l'optimisation de ces techniques pour améliorer à la fois la qualité sonore et l'efficacité de la compression. Le format IRM proposé dans ce projet représente une tentative de combiner les avantages des différentes méthodes de compression pour offrir une solution plus performante.

Ce projet vise à fournir une compréhension approfondie des différentes techniques de compression audio, à travers une étude théorique et pratique des formats couramment utilisés. En proposant une nouvelle méthode de compression innovante, ce projet contribue à l'amélioration continue des technologies audios numériques et à l'optimisation de la gestion des fichiers audio.

# Etude théorique :

## **Principe de compression audio avec perte et sans perte :**

La compression audio est une technologie fondamentale dans le domaine du traitement numérique du son. Elle vise à réduire la taille des fichiers audio tout en conservant leur qualité sonore selon les besoins spécifiques. Voici une explication approfondie des deux principaux types de compression : avec perte et sans perte.

Les fichiers audios, lorsqu'ils sont enregistrés dans leur forme brute, contiennent une grande quantité de données (exemple : un fichier WAV non compressé). Ces fichiers, bien qu'offrant une qualité sonore irréprochable, nécessitent un stockage et une bande passante importants. La compression audio permet de réduire la taille des fichiers pour :

* Faciliter leur stockage sur les appareils électroniques.
* Accélérer leur transmission via les réseaux (streaming, téléchargement).
* Optimiser l’utilisation de ressources matérielles.

Il existe deux types de compression :

* Compression avec perte
* Compression sans perte.

### **Compression avec perte :**

1. **Principe :**

La compression avec perte repose sur des algorithmes qui exploitent les limites de l'audition humaine. Certaines données, jugées imperceptibles, sont éliminées de manière irréversible pour réduire la taille du fichier.

1. **Étapes Clés :**
   * + Analyse Psychoacoustique : Identification des fréquences inaudibles ou masquées par d'autres sons.
     + Suppression des Données Non Essentielles : Élimination des parties du signal jugées inutiles.
     + Encodage : Application de techniques comme la transformation de Fourier ou le codage perceptuel**.**
2. **Exemples des formats :**

* OGG Vorbis : Format libre souvent utilisé pour le streaming.
* MP3 (MPEG Audio Layer 3) : L'un des formats les plus populaires.
* AAC (Advanced Audio Coding) : Utilisé par les services comme Spotify ou Apple Music.

1. **Les avantages :**

* Taille des fichiers considérablement réduite.
* Parfait pour les plateformes de streaming et les appareils à espace limité.

1. **Les inconvénients :**

* Perte définitive d'une partie des données.
* Qualité sonore inférieure, particulièrement perceptible avec du matériel audio haut de gamme.

### **Compression sans perte :**

1. **Principe :**

La compression sans perte repose sur des algorithmes qui réorganisent les données pour diminuer la taille des fichiers, sans perdre aucune information. Lors de la décompression, le fichier restauré est identique au fichier original.

1. **Étapes Clés :**

* **Analyse des Données :** Identification des redondances dans les séquences audios.
* **Codage :** Application d'algorithmes spécifiques comme :
  + *Codage Huffman* **:** Technique de compression basée sur la fréquence d'apparition des données.
  + *Codage LZW (Lempel-Ziv-Welch)* **:** Compression des chaînes répétitives dans les données.
* **Stockage Compressé :** Génération d'un fichier compressé tout en conservant sa qualité sonore intacte.

1. **Exemples des Format :**

* **FLAC (Free Lossless Audio Codec) :** Format ouvert et universel.
* **ALAC (Apple Lossless Audio Codec) :** Développé par Apple pour les appareils iOS.
* **AIFF (Audio Interchange File Format) :** Utilisé souvent pour l’édition musicale.

1. **Les avantages :**

* Qualité sonore parfaite.
* Idéal pour l'archivage et les environnements professionnels (studios).

1. **Les inconvénients :**

* Taille des fichiers plus volumineuse par rapport à la compression avec perte.
* Moins adapté pour le streaming ou le partage rapide.

### **Comparaison des Deux Types de Compression :**

|  |  |  |
| --- | --- | --- |
| **Critères** | **Compression sans perte** | **Compression avec perte** |
| Qualité Audio | Identique à l’original | Dégradée en fonction du taux de compression |
| Taille des Fichiers | Grande | Réduite |
| Applications | Archivage, Production musicale | Streaming, partages en ligne |
| Exemples | ALAC, FLAC, AIFF | OGG Vorbis, MP3, AAC |
| Flexibilité du Bitrate | Fixe | Variable (entre 96-320 kbps) |

La compression sans perte est privilégiée lorsqu'une qualité parfaite est requise (archivage, production). En revanche, la compression avec perte s'impose dans les situations où la taille réduite est essentielle (streaming, partages rapides). L’équilibre entre ces deux types de compression dépend des besoins spécifiques de l’utilisateur ou du service

## **Description des formats OGG et ALAC :**

Dans le domaine de la compression audio, chaque format a été conçu pour répondre à des besoins spécifiques, qu'il s'agisse de maximiser la qualité sonore ou d'optimiser le stockage et la transmission. Les formats **OGG** et **ALAC** représentent deux approches distinctes : l'une favorisant l'efficacité par une compression avec perte, et l'autre garantissant une qualité intacte grâce à une compression sans perte. Leur comparaison est essentielle pour comprendre leurs avantages, leurs inconvénients et leur impact dans des contextes variés, comme le streaming ou l'archivage.

Cette section vise à explorer les caractéristiques techniques des formats **OGG** et **ALAC**, en mettant en lumière leurs algorithmes, ainsi que leurs points forts et faibles. Ces détails permettront de mieux saisir les enjeux liés à leur adoption dans le cadre de la compression audio. Passons maintenant à la description détaillée de chaque format.

### **Format OGG :**

1. **Introduction :**

L’extension « .ogg » est associée au format de fichier OGG, qui est un conteneur multimédia (open source) développé par la fondation Xiph.Org(La Fondation Xiph.Org est une organisation à but non lucratif qui développe des formats et des logiciels open source pour la gestion des médias numériques (audio, vidéo, etc.). Il est principalement utilisé pour stocker de l'audio compressé, bien qu'il puisse aussi contenir de la vidéo, des sous-titres et d'autres types de données multimédias. Le format de conteneur OGG est un format multimédia qui peut contenir plusieurs flux de vidéo, audio, texte et métadonnées. Chaque flux peut être compressé avec différentes méthodes de compression, notamment :

* **Theora** (pour la vidéo)
* **Vorbis** (pour l’audio)
* **FLAC** (pour l’audio, compression sans perte)
* **OggPCM** (pour l’audio, non compressé)

1. **Principales caractéristiques et fonctionnalités :**
   * **Compression avec perte** : OGG Vorbis utilise un algorithme de compression perceptuelle, ce qui signifie qu’il supprime les données audios que l’oreille humaine perçoit moins, réduisant ainsi la taille du fichier tout en maintenant une bonne qualité sonore.
   * **Open source et libre de droits** : Contrairement au MP3 ou à l'AAC, OGG Vorbis est totalement exempt de brevets, ce qui le rend librement disponible pour les développeurs et créateurs de logiciels.
   * **Format conteneur** : Les fichiers OGG Vorbis sont généralement stockés dans un **format conteneur OGG** avec l'extension**.ogg**.

Le conteneur OGG peut également contenir d'autres types de médias, y compris de la vidéo.

* + **Profondeur de bits et fréquence d’échantillonnage**
  +  **Profondeur de bits** : Typiquement 16 bits, bien que le support du 24 bits existe, il est peu répandu.
  +  **Fréquence d’échantillonnage** : Supporte les fréquences standard (44,1 kHz, 48 kHz) et au-delà.
  + **Audio multicanal** : Prend en charge le son stéréo, mono et multicanal jusqu'à **255 canaux**, ce qui le rend adapté aux systèmes de son immersif.
  + **Décodage efficace** : Conçu pour un décodage rapide et efficace, ce qui en fait un bon choix pour le streaming et les appareils portables.

1. **Avantages :**

* **Qualité sonore supérieure à faible débit binaire** : Comparé au MP3, OGG Vorbis offre souvent une meilleure qualité sonore à débit binaire égal ou inférieur. Il est particulièrement efficace pour le streaming à faible débit
* **Open source et libre de droits** : Aucun frais de licence, ce qui en fait une option attrayante pour les développeurs de logiciels, les créateurs de jeux vidéo et les plateformes de streaming.
* **Encodage flexible du débit binaire** : Prend en charge **VBR, ABR et CBR**, permettant une meilleure optimisation de la taille du fichier et de la qualité.
* **Meilleure efficacité de compression** : OGG Vorbis peut produire des fichiers plus petits que le MP3 tout en conservant une qualité audio similaire ou meilleure.
* **Support du son multicanal** : Gère mieux l’audio immersif que le MP3.
* **Faible latence** : Utile pour les applications de streaming en temps réel.

1. **Inconvénients :**

* **Compatibilité matérielle limitée** : Contrairement au MP3, l’OGG Vorbis **n’est pas universellement pris en charge** sur tous les appareils matériels (ex. : autoradios plus anciens, certaines enceintes connectées et systèmes embarqués).
* **Problèmes de compatibilité** : Bien que la plupart des lecteurs logiciels modernes supportent OGG Vorbis,il reste moins universel que le MP3 et l’AAC.
* **Utilisation plus intensive du processeur pour l’encodage** : L’encodage des fichiers OGG Vorbis nécessite plus de puissance de calcul que le MP3, ce qui peut être un inconvénient pour certaines applications.
* **Limitations des métadonnées** : OGG Vorbis utilise un système de métadonnées spécifique (commentaires Vorbis) qui est moins standardisé que les balises ID3 utilisées par le MP3.

1. **Cas d’utilisation de l’OGG Vorbis :**

* **Streaming audio** : Grâce à son efficacité de compression, OGG Vorbis est souvent utilisé pour la radio en ligne, les applications VoIP et les bandes sonores de jeux vidéo.
* **Industrie du jeu vidéo** : De nombreux jeux (ex. : anciennes versions de Unreal Tournament et Quake) utilisent OGG Vorbis pour l’audio en jeu en raison de sa petite taille de fichier et de sa qualité correcte.
* **Projets open source** : Utilisé dans les lecteurs multimédias et plateformes open source grâce à l’absence de restrictions de licence.
* **Podcasting** : Certains podcasteurs utilisent OGG Vorbis pour éviter les formats propriétaires.
* **Archivage musical (lorsque la perte est acceptable)** : Lorsque la taille du fichier est un facteur important, mais qu’une bonne qualité audio est toujours souhaitée, OGG Vorbis est un bon choix.

1. **Compatibilité :**

* **Système d’exploitation :**
* **Windows** : Pris en charge par VLC, Foobar2000, Winamp et d'autres lecteurs tiers.
* **macOS** : Support natif limité, mais des lecteurs tiers comme VLC, Audacity et Clementine peuvent lire les fichiers OGG Vorbis.
* **Linux** : Pris en charge nativement sur la plupart des distributions.
* **Android** : De nombreuses applications de lecture musicale (ex : VLC, Poweramp, AIMP) prennent en charge OGG Vorbis.
* **iOS** : Support natif limité, mais VLC et d'autres applications tierces permettent la lecture des fichiers OGG Vorbis.
  + **Logiciels :**
* **Lecteurs multimédias :** VLC, Foobar2000, Winamp, Audacious, Clementine, et plus encore.
* ** Navigateurs Web** : Les navigateurs modernes (Chrome, Firefox et Edge) supportent OGG Vorbis pour la lecture audio en ligne.
* **Éditeurs audio :** Audacity, FFmpeg et d'autres outils de traitement audio prennent en charge OGG Vorbis.
  + **Matériel :**
    - Certains **baladeurs numériques** prennent en charge OGG Vorbis, mais ce n’est pas généralisé.
    - De nombreuses **consoles de jeux et appareils connectés** n’ont pas de support natif.
    - **Les autoradios et systèmes audio domestiques** peuvent ou non être compatibles, selon le fabricant.

1. **Algorithme de compression OGG Vorbis :**
2. **Prétraitement du signal audio dans la compression OGG Vorbis :**

Le prétraitement du signal est une étape essentielle dans l'algorithme de compression OGG Vorbis. Il permet de préparer le signal audio pour son analyse en fréquence et sa quantification. Cette phase comprend trois opérations principales :

1. **Charger le fichier audio brut (PCM) :**

Le format PCM (Pulse Code Modulation) est utilisé pour stocker un signal audio sous forme d’échantillons numériques. Chaque échantillon représente l’amplitude du signal à un instant donné.

**-**Le fichier audio est ouvert et converti en un tableau d’échantillons numériques.

**-** Ces échantillons sont organisés en valeurs entières ou flottantes (ex. 16 bits, 24 bits ou 32 bits).

**-** Un taux d’échantillonnage est défini (ex. 44.1 kHz ou 48 kHz pour l’audio standard)

1. **Diviser le signal en trames (blocs d'échantillons de 256 à 4096 échantillons) :**

Le signal audio est segmenté en blocs d'échantillons appelés trames (ou "frames"). Cela est nécessaire pour appliquer les transformations fréquentielles.

**Pourquoi cette segmentation ?**

• Un petit bloc (256 échantillons) capture mieux les hautes fréquences mais augmente le débit binaire.

• Un grand bloc (4096 échantillons) est plus efficace pour les basses fréquences mais a une latence plus élevée.

• OGG Vorbis adapte dynamiquement la taille des trames en fonction du contenu audio.

**Remarque :**

• Si le dernier bloc est plus court que la taille définie, on peut appliquer un remplissage (padding)

1. **Appliquer un fenêtrage (Hann, Vorbis ou autre) pour éviter les discontinuités aux bords des trames :**

Une fenêtre est une fonction mathématique appliquée à chaque trame pour adoucir les transitions entre les blocs. Sans cela, les discontinuités créeraient des artefacts audibles après la compression. Types de fenêtres utilisées en OGG Vorbis :

• Fenêtre de Hann (Hanning) – couramment utilisée en traitement du signal.

• Fenêtre de Vorbis – spécifique à OGG Vorbis, elle optimise la reconstruction du signal.

• Fenêtre de Hamming – similaire à Hann, mais avec des coefficients légèrement différents.

**Formule de la fenêtre de Hann** :

**Effet du fenêtrage :**

Évite les artefacts dus aux coupures abruptes.

Améliore la précision spectrale lors de la transformation en fréquence.

1. **Transformation en domaine fréquentiel (MDCT - Modified Discrete Cosine Transform) :**

Une fois le prétraitement terminé (segmentation en trames et application du fenêtrage), on applique la Transformée en Cosinus Discrète Modifiée (MDCT) pour convertir le signal du domaine temporel vers le domaine fréquentiel.

**Pourquoi utiliser la MDCT ?**

La MDCT est une variante de la Transformée en Cosinus Discrète (DCT) spécialement conçue pour les codecs audios avec pertes comme OGG Vorbis. Elle présente plusieurs avantages :

Meilleure répartition spectrale : réduit la redondance des données en analysant les fréquences. Fenêtrage avec recouvrement : réduit les artefacts liés à la discontinuité entre les trames successives.

Suppression des composantes inutiles : elle élimine certaines fréquences perceptuellement moins importantes.

La **MDCT** d`une trame de taille ***N*** produit ***N/2*** coefficients fréquentiels :

où :

• x(n) représente les échantillons de la trame après fenêtrage.

• N est la taille de la trame.

• k est l’indice fréquentiel des coefficients MDCT.

**Particularité :**

La MDCT est utilisée avec un recouvrement de 50% entre trames successives pour garantir une reconstruction parfaite du signal après compression

**Normalisation des coefficients MDCT** :

Une fois les coefficients calculés, on applique une normalisation pour optimiser leur quantification dans l'étape suivante.

• Les coefficients sont divisés par la somme des valeurs absolues pour éviter des écarts trop grands.

• Une quantification logarithmique peut être utilisée pour réduire la dynamique du signal.

1. **Masquage psychoacoustique :**

Le masquage psychoacoustique est une étape cruciale dans la compression audio avec pertes, comme dans le cas du codec OGG Vorbis. Il repose sur la façon dont l'oreille humaine perçoit les sons, permettant de réduire la quantité de données nécessaires pour encoder un signal tout en conservant une qualité sonore acceptable.

Le but de cette étape est de supprimer les informations audios inutiles, que l'oreille humaine ne peut pas percevoir en raison de phénomènes physiologiques, tels que la masse spectrale et la masking effect (effet de masquage).

1. **Principe du Masquage Psychoacoustique :**

Le masquage psychoacoustique repose sur deux phénomènes clés :

**-Masquage fréquentiel (ou spectral)** : Si deux sons sont proches en fréquence, un son plus fort peut rendre l'autre inaudible. Par exemple, un son de 1000 Hz peut masquer un son de 900 Hz s'il est suffisamment plus fort.

**-Masquage temporel** : Si un son intense est suivi rapidement par un autre son, il est possible que le second soit masqué, même s'il se situe à une fréquence différente. Cela est lié au phénomène de période de masquage, qui peut durer un certain temps après qu'un son fort a été joué.

Le masquage psychoacoustique dans le codec OGG Vorbis se sert de ces principes pour identifier les parties du signal audio qui sont inaudibles à l'oreille humaine et donc les supprimer ou les quantifier de manière plus agressive. Cela permet de réduire la taille du fichier sans que l'utilisateur n'entende de différences perceptibles.

1. **Étapes détaillées du Masquage Psychoacoustique :**

**-Calcul des spectres de puissance :**

* Après l'application de la MDCT (Modified Discrete Cosine Transform) sur chaque trame du signal, un spectre de puissance est obtenu pour chaque trame. Chaque coefficient MDCT représente une frequence du signal dans le domaine fréquentiel.

**-Identification des sons masquants :**

* Le modèle psychoacoustique évalue chaque coefficient MDCT pour déterminer si un son est susceptible de masquer un autre. Cela se fait en comparant l'intensité des différentes fréquences et en vérifiant si une fréquence plus forte est située à proximité d'une fréquence plus faible.
* Le seuil de masquage est calculé pour chaque fréquence. Ce seuil définit le niveau en dessous duquel un son est inaudible en raison du masquage par une fréquence plus intense. Il est souvent calculé à l'aide de courbes de masquage, qui sont basées sur les caractéristiques auditives de l'oreille humaine.

**-Application du seuil de masquage :**

* Une fois les seuils de masquage déterminés, tous les coefficients MDCT dont l'intensité est inférieure à ce seuil sont soit réduits, soit supprimés. Cela permet d'éliminer les composants audios inaudibles ou presque inaudibles. Par exemple, si un coefficient MDCT a une puissance trop faible pour être perçu par l'oreille humaine, il sera supprimé ou quantifié de manière plus grossière.

**-Calcul de la réduction de la bande passante :**

* Les harmoniques masquées sont supprimées ou codées plus efficacement, permettant de réduire la bande passante utilisée par ces parties du signal. Par conséquent, la compression devient plus efficace, avec une réduction significative du volume des données sans perte perceptible de qualité.

**-Optimisation du signal restant :**

* Après l'application du masquage, il peut être nécessaire d'ajuster la quantification des coefficients restants. Par exemple, on peut utiliser un codage plus agressif sur les coefficients restants qui sont jugés moins importants.

1. **Techniques de Masquage Utilisées dans OGG Vorbis :**

Le codec OGG Vorbis utilise plusieurs techniques spécifiques pour effectuer un masquage psychoacoustique :

**-Masquage fréquentiel** : Le modèle de Vorbis calcule des seuils de masquage pour chaque trame. Ces seuils sont basés sur la puissance spectrale du signal et sur la sensibilité de l'oreille humaine à certaines fréquences. Le seuil de masquage est appliqué sur les coefficients MDCT, éliminant ceux qui sont inaudibles.

**-Évaluation de la période de masking temporelle** : Après un son intense, certaines fréquences peuvent être masquées pendant une période donnée (temps). Cela est pris en compte lors du masquage des harmoniques suivant immédiatement un pic de fréquence.

**-Optimisation de l'encodage** : Après avoir supprimé ou masqué certaines fréquences, le codec Vorbis applique des techniques de quantification perceptuelle, où des informations supplémentaires sur la structure du signal (par exemple, la dynamique du signal) sont utilisées pour coder plus efficacement les coefficients restants.

1. **Résultats du Masquage Psychoacoustique :**

**-Suppression des fréquences inaudibles** : Les parties du signal qui sont masquées sont supprimées ou codées de manière moins précise, permettant une réduction de la taille du fichier audio tout en conservant la qualité perçue.

**-Amélioration de l'efficacité de la compression** : En réduisant ou éliminant les informations inaudibles, le codec peut utiliser des techniques de codage plus efficaces sur les coefficients restants, réduisant ainsi la taille globale du fichier sans nuire à la qualité sonore perçue.

### **Format ALAC :**

1. **Introduction :**

ALAC (Apple Lossless Audio Codec) est un format de compression audio sans perte développé par Apple Inc. Son objectif est de compresser l’audio numérique non compressé(généralement au format PCM, tel que WAV ou AIFF) sans aucune perte d’information, garantissant que l’audio décompressé est identique à l’original. Les avantages d’ALAC incluent un stockage efficace (réduction de la taille des fichiers à environ 40-60 % de l’original), une faible charge de calcul lors du décodage et une intégration transparente dans l’écosystème Apple (iTunes, iOS, macOS, etc.).

1. **Principales caractéristiques et fonctionnalités :**

* **Compression sans perte** : Contrairement aux codecs avec perte (comme MP3 ou AAC), ALAC ne supprime aucune donnée audio pendant la compression. Il utilise des techniques qui identifient et suppriment les redondances dans le signal audio sans sacrifier d’informations. Lors de la décompression, les données audios sont parfaitement reconstituées à leur état d'origine.
* **Open Source** : Fin 2011, Apple a rendu le décodeur et l'encodeur ALAC open source. Cela a permis une adoption plus large et une implémentation au-delà de l’écosystème Apple. Le code source est disponible sous licence Apache 2.0.
* **Format conteneur** : Les fichiers ALAC sont généralement stockés dans un conteneur MPEG-4 Part 14, avec l’extension (. M4A). Bien que .m4a soit souvent associé à l'AAC (avec perte), il peut également contenir de l’audio ALAC (sans perte). Parfois, l’extension ALAC est utilisée, mais (.M4A) reste la norme.
* **Profondeur de bits et fréquence d'échantillonnage prises en charge :**
* **Profondeur de bits :** 16 bits, 20 bits, 24 bits et même 32 bits en entier.
* **Fréquence d'échantillonnage :** De très basses fréquences jusqu'à 384 kHz

(et théoriquement au-delà, bien que des limites pratiques existent).

Cette large prise en charge permet à ALAC de gérer efficacement les enregistrements audio haute résolution.

* **Canaux :** Prise en charge du son mono, stéréo et multicanal (comme le son surround ).
* **Compatibilité matérielle et logicielle :**
* **Écosystème Apple :** Support natif sur tous les appareils Apple (macOS, iOS, iPadOS, tvOS, watchOS, HomePod). Intégré aux logiciels Apple comme iTunes (désormais l’application Musique), QuickTime et le framework Core Audio.
* **Support tiers** : Grâce à son statut open source, ALAC est pris en charge par divers lecteurs multimédias, logiciels d'édition audio et systèmes d'exploitation hors de l’écosystème Apple. Bien qu’il ne soit pas aussi universel que le MP3, son adoption est assez large.
* **Efficacité de compression :** ALAC reste un format compressé malgré son caractère sans perte. Le taux de compression se situe généralement entre 40 et 60 % de la taille du fichier non compressé original, ce qui est moins efficace qu'un codec avec perte, mais nettement mieux que les formats non compressés comme WAV ou AIFF.
* **Utilisation du processeur :** Le décodage de l’ALAC est généralement moins gourmand en ressources CPU que certains autres codecs sans perte comme le FLAC. Cela peut être un avantage sur des appareils ayant une puissance de traitement limitée ou pour préserver l’autonomie des batteries

1. **Avantages :**

* **Qualité audio sans perte** **:** Le principal avantage d’ALAC est qu’il préserve toute la dynamique et les détails de l’enregistrement original, ce qui le rend idéal pour les audiophiles et l’archivage.
* **Open Source :** Sa disponibilité en open source favorise une adoption plus large, une meilleure interopérabilité et des améliorations par la communauté.
* **Intégration à l’écosystème Apple :** Un support natif et transparent dans l’environnement Apple en fait un choix privilégié pour les utilisateurs Apple.
* **Bon taux de compression pour un format sans perte :** Un bon équilibre entre qualité audio et taille de fichier comparé aux formats non compressés.
* **Faible consommation CPU :** Un décodage efficace, bénéfique pour les appareils mobiles et les anciens matériels.
* **Support de l’audio haute résolution :** Compatible avec les enregistrements en haute définition.

1. **Inconvénients :**

* **Taille des fichiers :** La compression sans perte entraîne des fichiers bien plus volumineux que les formats avec perte comme MP3 ou AAC.
* **Compatibilité (historique) :** Bien qu'il soit open source aujourd’hui, ALAC était historiquement lié à Apple. Son support s'est élargi, mais il reste moins universel que le MP3, en particulier sur d’anciens appareils ou plateformes de niche.
* **Taux de compression inférieur aux formats avec perte :** Pour des situations où l’espace de stockage est critique et que la fidélité audio n’est pas une priorité, des formats comme MP3 ou AAC sont préférables.

1. **Cas d’utilisation de ALAC :**

* **Archivage audio :** Idéal pour conserver des enregistrements précieux tout en préservant leur qualité originale.
* **Audiophiles et écoute haute fidélité :** Permet une restitution fidèle du son, exactement comme l’artiste l’a voulu
* **Utilisateurs de l’écosystème Apple :** Solution naturelle et bien intégrée pour stocker et écouter de la musique en haute qualité.
* **Édition et production audio :** Utilisé dans certains flux de travail nécessitant une qualité sans perte, bien que des formats non compressés comme WAV ou AIFF soient plus courants dans les studios professionnels.
* **Systèmes audio domestiques :** Utilisé dans les systèmes de diffusion musicale haute fidélité.

1. **Compatibilité :**

* **Systèmes d’exploitation :**
  + - * **macOS :** Support natif.
      * **iOS/iPadOS :** Support natif.
      * **Windows :** Supporté par iTunes, VLC, Foobar2000, etc.
      * **Linux :** Supporté par VLC, Foobar2000 (via Wine), Audacious, Clementine, etc.
      * **Android :** Support variable selon les applications (VLC, Poweramp, etc.).
* **Logiciels :**
* **Apple Music (iTunes), QuickTime Player, VLC, Foobar2000,** **Audacity,** et d’autres lecteurs et éditeurs audio.
* **Matériel :**
* **Appareils Apple :** iPhones, iPads, Macs, Apple TVs, HomePods (support natif).
* **Lecteurs Hi-Fi** et **serveurs audio réseau** prenant en charge l’ALAC.

1. **Algorithme de compression ALAC :**
2. **Segmentation en blocs:**

Le processus commence par la division du signal audio en **blocs de données**.

Cette segmentation permet :

* De traiter le signal par sections pour optimiser les calculs.
* De s’adapter aux variations locales du signal en ajustant les paramètres de compression pour chaque bloc.

Chaque bloc peut varier en taille selon l’implémentation, mais il est conçu pour équilibrer efficacité et précision.

1. **Décorrélation des canaux (si stéréo ou multicanaux) :**

Si l'audio contient plusieurs canaux (comme en stéréo), une **transformation de prédiction** est appliquée pour réduire la redondance entre les canaux :

* **Mode indépendant** : Les canaux sont traités séparément.
* **Mid/Side (M/S)** : Transformation des signaux en composantes communes (**Mid**) et différences (**Side**).
* **Différentiel** : Calcul des différences entre les canaux corrélés.

1. **Prédiction linéaire et soustraction :**

L'idée principale est de **prédire** la valeur d'un échantillon à partir des valeurs précédentes et d’encoder **uniquement l'erreur de prédiction**. Cela réduit la quantité d'informations à stocker.

1. **: Pourquoi utiliser la prédiction linéaire ?**

Les signaux audios sont souvent **corrélés** :

* Un échantillon est généralement **proche** du précédent.
* En utilisant un **modèle mathématique de prédiction**, on peut deviner la prochaine valeur avec une **bonne précision**.
* L’erreur (résidu) est souvent **faible**, ce qui améliore la compression.

L’objectif est donc de stocker **uniquement les différences (erreurs)** entre la valeur réelle et la valeur prédite.

1. **Formule générale de la prédiction linéaire**

L’estimation d’un échantillon audio est donnée par une combinaison linéaire des **k derniers échantillons** :

Tel que :

* S(n) est la **valeur réelle** de l’échantillon.
* S′(n) est la **valeur prédite**.
* a1,a2,...,ak sont des **coefficients optimisés** en fonction du signal.

En pratique, ALAC utilise une **prédiction basée sur 1 ou 2 échantillons précédents**.

1. **Calcul du résidu :**

Ensuite on calcule l’erreur de prédiction en utilisant la relation suivante :

Les résidus sont généralement de petites valeurs autour de zéro.

1. **Codage Rice (Golomb-Rice) :**

Une fois le résidu de prédiction calculé, ALAC utilise le codage Rice pour encoder ces valeurs efficacement. Ce codage est adapté aux données avec une distribution géométrique ou quasi-géométrique (courante dans les résidus de prédiction). Il attribue des codes courts aux résidus fréquents et de faible amplitude, et des codes plus longs aux résidus rares et de grande amplitude. Le paramètre Rice est sélectionné de manière adaptative pour chaque bloc afin de correspondre aux propriétés statistiques des résidus, atteignant ainsi la limite d’entropie théorique.

1. **Codage par longueur de séquence (RLE) :**

Dans certains cas, notamment lorsque la prédiction est très efficace, les résidus peuvent contenir de longues séquences de valeurs identiques (souvent des zéros). Le codage RLE compresse ces séquences en les représentant par une paire indiquant la valeur et le nombre de répétitions, au lieu de stocker chaque valeur individuellement. Cette étape réduit encore plus la redondance dans les segments d’audio hautement prévisibles.

Lorsque la prédiction est très efficace, il peut y avoir de longues séquences de résidus identiques (souvent des zéros).

**Compression :**

* + Une séquence de N zéros est représentée par une paire (0,N).
  + Cette méthode réduit considérablement la taille des segments répétitifs.

**Avantage :** Réduction significative des données lorsque le signal est prévisible.

1. **Principes de codage entropique (similaires au codage Huffman) :**

Bien qu’ALAC n’utilise pas typiquement le codage Huffman classique, il suit le même principe sous-jacent : attribuer des codes de longueur variable en fonction de la probabilité des données. Les combinaisons de prédiction linéaire et de codage Rice adaptatif permettent d’assurer que les résidus les plus probables (ou les plus petits) sont encodés avec moins de bits, obtenant ainsi une compression sans perte efficace.

1. **Encapsulation dans un conteneur M4A :**

Les données compressées sont stockées dans un **fichier M4A** avec les métadonnées nécessaires.

**Pourquoi “M4A”?**

* Compatible avec iTunes, iOS et macOS.
* Supporte les métadonnées (nom de l’artiste, album, genre, etc.).
* Facile à lire par de nombreux logiciels audios.

### **Comparaison générale :**

|  |  |  |
| --- | --- | --- |
| **Critère** | **OGG** | **ALAC** |
| **Compression** | Avec perte | Sans perte |
| **Qualité audio** | Bonne | Excellente(originale) |
| **Taille du fichier** | Plus petite | Plus grande |
| **Licence** | Libre/Open source | Open source (Apple) |
| **Compatibilité** | Android, Linux, Web | Apple (iOS, macOS) |
| **Usage typique** | Streaming, Web | Archivage, hi-fi Apple |

# Extension personnalisée « .irm »:

## **Analyse et planification :**

### **Introduction à l'extension IRM :**

Dans ce chapitre, nous présentons notre innovation principale : le développement d'un nouveau format de compression audio nommé IRM. Cette proposition s'inscrit dans la lignée de notre étude comparative des formats existants, tout en cherchant à apporter une solution qui répond aux limitations identifiées dans notre analyse. Le format IRM a été conçu pour offrir un équilibre optimal entre qualité audio et taux de compression, en combinant efficacement plusieurs algorithmes de codage.

### **Contexte et objectifs du format IRM**

Le développement du format IRM s'inscrit dans le cadre du mini-projet de Traitement des Données Multimédia (TDM) de la Licence Informatique, Réseaux et Multimédia (IRM) à la Faculté des Sciences et Techniques de Mohammedia. Notre objectif était de proposer une alternative innovante aux formats de compression audio existants, en exploitant les connaissances acquises en matière de codage et compression numérique.

Le format IRM vise à :

* Réduire efficacement la taille des fichiers audio sans compromettre la qualité sonore
* Proposer une alternative aux formats commerciaux dominants.
* Exploiter les avantages des algorithmes de compression vus en cours (Huffman, LZW)
* Offrir une flexibilité d'utilisation adaptée aux besoins des utilisateurs

### **Présentation de la problématique :**

Malgré l'existence de nombreux formats de compression audio, chacun présente des avantages et des inconvénients spécifiques :

* Les formats sans perte (FLAC, ALAC) préservent la qualité mais offrent des taux de compression modérés
* Les formats avec perte (MP3, AAC) atteignent des taux de compression élevés mais au détriment de la qualité
* Certains formats sont propriétaires, limitant leur utilisation dans des projets open source
* La plupart des solutions n'offrent pas d'adaptabilité en fonction du type de contenu audio

Notre problématique consiste donc à développer un nouveau format qui tente de concilier ces différentes contraintes tout en proposant une approche innovante basée sur la combinaison de plusieurs algorithmes de compression.

### **Étude de l'existant :**

On a réalisé une étude approfondie des formats qui nous ont été assignés, notamment OGG et ALAC. Cette analyse a porté sur :

* **Architecture des formats** : Structure des fichiers, organisation des métadonnées
* **Algorithmes utilisés** : Techniques de prédiction, codage entropique
* **Performance** : Taux de compression, vitesse d'encodage/décodage
* **Qualité** : Fidélité au signal original, artefacts audibles
* **Compatibilité** : Support par les différents systèmes et applications

Cette étude a révélé que :

* OGG (avec le codec Vorbis) utilise une compression avec pertes basée sur la transformée en cosinus discrète modifiée
* ALAC (Apple Lossless Audio Codec) emploie une compression sans perte avec prédiction linéaire et codage entropique
* Chaque format présente des forces et des faiblesses selon le type de contenu audio

### **Solution proposée :**

Suite à cette analyse, on a proposé le développement du format IRM qui :

* Combine trois techniques complémentaires (Huffman, LZW et masquage psychoacoustique)
* Propose une entête optimisée permettant de stocker des métadonnées essentielles
* Offre une interface graphique intuitive pour l'utilisateur final

### **Cahier des charges**

1. **Les clients cibles par notre interface graphique :**

* Professionnels de l'audio (musiciens, ingénieurs du son)
* Développeurs d'applications multimédia
* Utilisateurs avertis cherchant des alternatives aux formats standards
* Établissements universitaires et de recherche

1. **Objectifs de l'interface graphique :**

* Concevoir un format de compression audio innovant
* Implémenter les algorithmes RLE, Huffman et LZW pour la compression audio
* Développer une interface utilisateur permettant de compresser/décompresser des fichiers
* Fournir des outils de mesure des performances (taux de compression, qualité)
* Comparer objectivement les résultats avec les formats existants

1. **Description des besoins fonctionnels**

* **Gestion des fichiers audio** :
  + Importer des fichiers audios dans différents formats
  + Exporter les fichiers compressés au format IRM
  + Convertir entre différents formats
* **Compression et décompression** :
  + Compresser un fichier audio au format IRM
  + Décompresser un fichier IRM vers un format standard
  + Ajuster les paramètres de compression
* **Analyse et comparaison** :
  + Afficher les caractéristiques des fichiers audio
  + Mesurer le taux de compression
  + Évaluer la qualité audio après compression
  + Comparer les performances avec d'autres formats
* **Interface utilisateur** :
  + Afficher une interface graphique intuitive
  + Visualiser des représentations du signal audio
  + Fournir des retours visuels sur les processus de compression

1. **Description des besoins non fonctionnels :**

* **Performance** :
  + Temps de compression/décompression raisonnable
  + Utilisation optimisée des ressources système
* **Fiabilité** :
  + Gestion robuste des erreurs
  + Préservation de l'intégrité des données
* **Utilisabilité** :
  + Interface intuitive et accessible
  + Documentation claire pour l'utilisateur
* **Maintenabilité** :
  + Code modulaire et bien documenté
  + Architecture extensible
* **Portabilité** :
  + Compatibilité avec différents systèmes d'exploitation
  + Dépendances minimales

1. **Contraintes et limitations :**

* Implémentation réalisée en Python
* Nécessité de bibliothèques spécifiques pour le traitement audio
* Performances limitées par l'utilisation de Python (par rapport à des langages compilés)
* Temps de développement contraint par le calendrier académique

## **Réalisation de l’extension personnalisée « .irm »:**

### **Introduction :**

Ce chapitre présente en détail la mise en œuvre du format de compression audio IRM et le développement de l'application associée. Il décrit l'architecture technique, les fonctionnalités implémentées et les résultats obtenus lors des tests de performance.

### **Format IRM : Architecture et caractéristiques :**

Le format IRM a été conçu pour offrir un compromis optimal entre taux de compression et qualité audio. Sa structure se compose de :

* Entête du fichier.
* Données audio compressée.

1. **En-tête du fichier « irm » :**

L'en-tête du format de compression audio IRM a été conçu pour être à la fois compact et complet, fournissant toutes les informations nécessaires à la décompression tout en minimisant l'overhead. Voici une analyse détaillée de sa structure :

**Signature (4 octets) :**

La signature "IRM\0" (49 52 4D 00 en hexadécimal) permet d'identifier immédiatement le format du fichier et de vérifier son intégrité lors de la lecture.

**Version (2 octets) :**

Le format de version utilise deux octets distincts pour représenter les numéros de version majeure et mineure, permettant une évolution compatible du format :

* Premier octet : numéro de version majeure
* Second octet : numéro de version mineure

Par exemple, pour la version 1.0, ces octets contiennent respectivement les valeurs 1 et 0.

**Type de compression (1 octet)**

Ce champ encode la stratégie de compression utilisée selon la correspondance suivante :

* **0: "LZW "** - Compression LZW uniquement
* **1 : "Huffman "** - Codage de Huffman uniquement
* **2 : "Huffman + LZW**" - Combinaison séquentielle des deux méthodes
* **3 : "Huffman + LZW + Masquage"** - Utilisation des trois techniques incluant le masquage perceptuel
* **4 : "Huffman + Masquage"** - Codage de Huffman avec masquage perceptuel
* **5 : "LZW + Masquage"** - Compression LZW avec masquage perceptuel

**Fréquence d'échantillonnage (4 octets)**

Stockée en format little-endian sur 4 octets, cette valeur représente la fréquence d'échantillonnage en Hertz (par exemple 44100 Hz pour qualité CD). Ces 4 octets permettent de stocker des valeurs allant jusqu'à 4,29 GHz, largement suffisant pour toutes les applications audios actuelles et futures.

**Taille de fenêtre MDCT (2 octets)**

Ce paramètre, encodé en little-endian sur 2 octets, indique la taille de fenêtre utilisée pour la transformée MDCT (Modified Discrete Cosine Transform). Cette information est cruciale pour reconstruire correctement le signal audio lors de la décompression.

**Dimensions des données (8 octets)**

Ce champ contient la forme de la matrice de coefficients MDCT, décomposée en :

* 4 octets (little-endian) pour le nombre de lignes
* 4 octets (little-endian) pour le nombre de colonnes

Cette information permet de reformer correctement la matrice de coefficients à partir des données compressées lors de la phase de décompression.

1. **Données audio compressées :**

La section des données audio compressées constitue le cœur du fichier IRM et contient l'information sonore encodée selon la méthode spécifiée dans l'en-tête. Sa structure varie en fonction de l'algorithme de compression choisi, mais présente les caractéristiques suivantes :

**Structure générale :**

* Un flux binaire optimisé dont l'organisation dépend directement du type de compression utilisé (LZW, Huffman, ou leurs combinaisons).
* Pour les méthodes utilisant Huffman, la table de codage est stockée au début de cette section pour permettre le décodage.
* Dans le cas de LZW, le dictionnaire initial est implicite et n'est pas stocké, mais les indices sont encodés avec une taille variable optimisée.

### **Implémentation des algorithmes de compression**

L’architecture du format «.irm » repose sur une chaîne de traitement audio modulaire, structurée en plusieurs étapes :

**1. Prétraitement du signal** (centrage et normalisation)

**2. Tramage** (division en blocs temporels)

**3. Fenêtrage et overlap** (pour la continuité du signal)

**4. Transformée FFT** (accélération du calcul fréquentiel et analyse)

**5. Transformée MDCT** (représentation fréquentielle compacte)

**6. Analyse psychoacoustique** (facultative selon l’algorithme choisi)

**7. Requantification adaptative**

8. Application de l’un des algorithmes de compression suivants :

○ Huffman seul (sans perte)

○ LZW seul (sans perte)

○ Huffman + masquage psychoacoustique (avec perte)

○ LZW + masquage psychoacoustique (avec perte)

○ Huffman + LZW + masquage psychoacoustique (avec perte combinée)

Chaque étape a été choisie pour répondre à un objectif précis dans le processus global de compression :

● **Le prétraitement** (centrage et normalisation) prépare le signal à être analysé mathématiquement avec précision, en supprimant les biais et en le ramenant dans une plage standardisée.

● **Le tramage** permet un traitement localisé du signal pour capter sa variation temporelle. ● Le fenêtrage et le recouvrement (overlap) assurent la continuité entre trames et réduisent les discontinuités à la reconstruction.

● **La FFT**, utilisée ici en pré-analyse, permet une lecture rapide du contenu fréquentiel utile pour ajuster dynamiquement certains paramètres de compression, notamment pour accélérer ou calibrer la MDCT sans devoir recourir systématiquement à l’analyse psychoacoustique.

● **La MDCT** transforme le signal temporel en une représentation fréquentielle efficace à coefficients réels, idéale pour la requantification et la réduction de données.

● **L’analyse psychoacoustique**, lorsqu’elle est activée, identifie les zones inaudibles à supprimer ou compresser plus fortement, pour optimiser la compression avec perte.

● **La requantification** permet d’adapter la précision des coefficients MDCT selon leur importance auditive.

● Enfin, **les algorithmes de compression Huffman et LZW**, seuls ou combinés avec du masquage, permettent de coder les données de façon efficace, en minimisant la taille finale du fichier.

1. **Le prétraitement :**

Un fichier audio brut contient des échantillons numériques représentant les variations de pression acoustique. Ces échantillons sont souvent stockés sous forme d'entiers (ex : int16). Avant de travailler sur ce signal (FFT, psychoacoustique, quantification, etc.), on doit le préparer proprement pour éviter des erreurs de traitement ou des biais mathématiques. Deux étapes clés :

● Centrage : supprimer un éventuel décalage DC (offset).

● Normalisation : adapter les valeurs à une échelle standard (souvent entre

-1 et 1).

**Le centrage :**

Le centrage est une étape fondamentale du prétraitement du signal audio. Elle consiste à supprimer la composante continue (appelée décalage DC, pour Direct Current) en recentrant le signal autour de l’axe zéro.

Lors de l’acquisition audio (par exemple via un microphone), les imperfections électroniques des capteurs ou interfaces de conversion analogique/numérique peuvent introduire un décalage vertical du signal. Ce décalage se manifeste par une moyenne non nulle des échantillons, c’est-à-dire une composante continue parasite. Cette anomalie est appelée offset DC (DC offset). Un signal audio sain doit osciller autour de zéro, avec une distribution équilibrée de valeurs positives et négatives. Le centrage permet donc de rétablir cette propriété physique essentielle en supprimant l’offset DC.

**La normalisation :**

Une fois le signal centré, on applique une normalisation pour que l’amplitude maximale soit limitée à une valeur de référence, en général Cela permet d’éviter la saturation, de rendre le signal plus cohérent, et d’optimiser les traitements numériques suivants comme la quantification ou la FFT. La normalisation se fait simplement en divisant tous les échantillons par la valeur maximale absolue du signal.

La normalisation est une opération de mise à l’échelle du signal audio, effectuée après le centrage, qui vise à ajuster l’amplitude maximale du signal à une valeur cible, généralement 1 ou -1 dans le cas des signaux normalisés flottants (en float32 ou float64).

La normalisation permet donc de :

● Maximiser l’utilisation de la plage dynamique disponible,

● Uniformiser les signaux issus de différentes sources,

● Préparer le signal pour l’analyse fréquentielle, la quantification ou la compression.

L’objectif de la normalisation est multiple :

**1.** Adapter le signal à la plage dynamique admissible du système numérique ([-1, 1] pour les formats en virgule flottante, [−32768, 32767] pour les entiers 16 bits, etc.).

**2.** Prévenir la saturation (clipping) ou l’écrasement du signal.

**3.** Harmoniser les signaux provenant de différentes sources (voix, instruments, bruit, etc.).

**4.** Préparer le signal pour les étapes suivantes du traitement : analyse spectrale (FFT), quantification, compression.

La normalisation d’un signal audio consiste à ajuster l’amplitude maximale du signal afin de l’inscrire dans une plage prédéfinie, sans modifier la forme du signal. Cette opération linéaire garantit la compatibilité du signal avec les traitements numériques, évite les saturations et assure une base saine pour la compression, la quantification ou l’analyse spectrale. Elle constitue une étape incontournable dans tout pipeline de traitement audio avec pertes.

1. **Le tramage du signal audio :**

Le tramage consiste à découper le signal audio en segments courts successifs appelés trames (frames). Chaque trame contient un petit nombre d’échantillons (souvent entre 256 et 2048) et correspond à une portion très courte du signal dans le temps (généralement entre 10 ms et 50 ms).

Le tramage est indispensable car :

● Le signal audio change dans le temps,

● Les méthodes comme la transformée de Fourier (FFT) supposent que le signal est stationnaire (c’est-à-dire que ses caractéristiques ne changent pas),

● En le découpant en petits morceaux, on peut supposer qu’il reste à peu près stationnaire pendant une trame.

1. **Overlap (Chevauchement des trames) :**

Afin de préparer le signal à une transformation fréquentielle efficace, le premier traitement appliqué est le chevauchement temporel des trames, également appelé overlap. Ce procédé consiste à diviser le signal en blocs de 2N2N2N échantillons, avec un décalage de NNN échantillons entre deux trames consécutives. Cela implique que chaque trame chevauche la moitié de la précédente et de la suivante. Le chevauchement permet de minimiser les discontinuités entre les trames successives, en particulier au niveau des frontières, réduisant ainsi les artefacts perceptibles à la reconstruction. Il s'agit également d'une condition indispensable à l'application correcte de la transformée MDCT et de son inverse (IMDCT), dans le cadre du schéma overlap-add permettant la reconstruction sans erreur du signal (hors pertes dues à la compression). Ce chevauchement est notamment utilisé dans les standards de compression audio modernes tels que MP3 ou AAC.

1. **Fenêtrage (Windowing) :**

Une fois les trames obtenues via le chevauchement, chacune d’elles subit une opération de fenêtrage. Le but du fenêtrage est d’atténuer progressivement les valeurs du signal aux extrémités des trames, évitant ainsi des transitions abruptes susceptibles d'introduire des artefacts spectro-temporels. Pour cela, chaque trame est multipliée par une fonction appelée fenêtre, généralement de forme sinusoïdale, et répondant à des critères stricts. Dans notre cas, nous utilisons la fenêtre de Princen-Bradley, qui garantit la reconstruction parfaite lors du recouvrement des trames inversées (IMDCT).

1. **MDCT : Transformée en Cosinus Discrète Modifiée :**

La MDCT est une transformation orthogonale dérivée de la DCT-IV, conçue pour les signaux à trames chevauchées. Elle transforme un bloc temporel de 2N échantillons en N coefficients fréquentiels réels, permettant une représentation compacte et adaptée à la psychoacoustique

La MDCT présente plusieurs propriétés qui en font un outil central dans les algorithmes de compression audio modernes. D'abord, elle permet de réduire significativement la taille des données, en transformant un bloc de signal en deux fois moins de coefficients fréquentiels. Cette réduction facilite considérablement les étapes de traitement et de stockage. De plus, la MDCT fournit une représentation fréquentielle optimisée pour les besoins de l'analyse psychoacoustique. Cela signifie que l'information utile à l’audition humaine est mieux mise en valeur, ce qui permet de cibler plus efficacement les parties du signal pouvant être éliminées ou simplifiées sans dégradation audible. Autre avantage majeur : la MDCT produit uniquement des coefficients réels, ce qui simplifie les opérations de quantification et de codage. Contrairement à la FFT, elle n'implique pas de parties imaginaires, ce qui réduit la complexité du traitement et la taille des données. Enfin, la MDCT est parfaitement compatible avec le schéma de reconstruction par recouvrement-addition. Lorsqu’elle est utilisée avec un chevauchement de 50 % entre trames et un fenêtrage adapté (comme la fenêtre sinusoïdale de Princen-Bradley), elle permet une reconstruction exacte du signal d’origine, à condition que la quantification ne soit pas encore appliquée. La reconstruction se fait en trois étapes : l’application de la transformée inverse (IMDCT), la superposition des trames (overlap), puis l’addition des portions communes. Ce mécanisme garantit une transition fluide et sans discontinuités entre les trames, assurant ainsi une restitution audio de haute qualité.

1. **Modélisation psychoacoustique en compression audio:**

La psychoacoustique est la branche de la psychophysique qui étudie la perception des sons par l’être humain. Elle ne s’intéresse pas uniquement aux caractéristiques physiques du son (comme l’amplitude ou la fréquence), mais à la manière dont notre cerveau interprète ces sons. Cela inclut :

● les seuils d’audibilité,

● les effets de masquage,

● la sensibilité de l’oreille humaine selon la fréquence,

● la résolution temporelle et fréquentielle de l’audition.

Dans le contexte de la compression audio, la psychoacoustique est utilisée pour identifier les parties du signal que l’auditeur ne perçoit pas ou perçoit très mal, afin de les supprimer ou de les simplifier sans altérer la qualité perçue. On parle alors de réduction de la redondance perceptuelle

**Pourquoi utiliser la psychoacoustique dans la compression avec perte ?**

L’objectif d’un codec avec perte (comme .irm) est de réduire la taille des données audio, tout en maintenant une qualité perçue optimale.

Grâce à la psychoacoustique, il est possible de :

● Supprimer des informations inaudibles (donc inutiles à stocker),

● Réduire la précision de certains échantillons sans que cela ne soit remarqué,

● Allouer plus de bits aux sons facilement perçus, et moins aux sons masqués ou marginaux.

**-Seuil d’audibilité (Absolute Threshold of Hearing) :**

Le seuil d’audibilité est la plus faible intensité sonore qu’un être humain peut percevoir, selon la fréquence. Il n’est pas plat : l’oreille humaine est la plus sensible autour de 2 kHz à 5 kHz (voix humaine, bruits d’alerte), et beaucoup moins en dessous de 100 Hz ou au-dessus de 15 kHz.

La courbe du seuil d’audibilité est décrite par les courbes de Fletcher-Munson (ou ISO 226), qui montrent les intensités nécessaires à différentes fréquences pour qu’un son soit perçu comme ayant le même volume. En compression, on compare les niveaux de nos coefficients MDCT à ce seuil : s’ils sont en dessous, ils peuvent être supprimés.

**-Masquage auditif (Auditory Masking) :**

▶ **Masquage spectral (fréquentiel)**

Lorsqu’un son est fort dans une bande de fréquence, il peut masquer les sons plus faibles dans ses fréquences voisines. Par exemple, si une basse de 100 Hz est très forte, un bruit léger à 120 Hz devient inaudible. L’oreille ne peut pas distinguer les deux. On peut donc supprimer les signaux masqués. Le masquage spectral est modélisé par des "masques en cloche" centrés sur les pics d’énergie du spectre. On évalue pour chaque trame MDCT les zones où des coefficients sont masqués par d'autres.

▶ **Masquage temporel :** Un son fort peut également masquer un son qui arrive juste avant ou juste après lui dans le temps :

● Masquage avant (pre-masking) : quelques millisecondes avant un son fort.

● Masquage après (post-masking) : jusqu'à 100 ms après un son fort.

C’est un phénomène utile car il permet de simplifier les transitions rapides dans l’audio sans les rendre audibles.

**-Bande critique et banc de filtres (Bark scale)**

Le système auditif humain ne traite pas les fréquences de manière linéaire, mais en bandes critiques (environ 24 chez l'humain), correspondant à la plage de résolution fréquentielle de l’oreille. Ces bandes sont modélisées par l’échelle de Bark, qui regroupe les fréquences en zones auditivement homogènes.

Lorsqu’on applique la MDCT, on peut regrouper les coefficients par bandes de Bark pour analyser l’énergie perceptible dans chaque bande. Cela permet de définir des niveaux de masquage spécifiques par bande, et de quantifier les coefficients en fonction de leur perceptibilité

1. **Quantification et Requantification des coefficients MDCT :**

La quantification est le processus qui consiste à réduire la précision des valeurs numériques, c’est-à-dire remplacer une valeur réelle (à virgule flottante) par une valeur discrète (entier), appartenant à un ensemble fini de niveaux. En compression audio, cela signifie qu’on remplace les coefficients MDCT (souvent en virgule flottante) par des valeurs entières approximées, ce qui permet de :

● réduire le nombre de bits nécessaires pour les stocker.

● faciliter le codage entropique (comme Huffman).

**Pourquoi on parle de requantification ?**

On parle souvent de requantification plutôt que quantification tout court, car :

● les signaux audios numériques sont déjà quantifiés une première fois lors de leur enregistrement (ex : 16 bits par échantillon),

● la requantification ici intervient après transformation MDCT et masquage psychoacoustique : on réduit la précision une deuxième fois, de manière adaptative, en tenant compte de la perception auditive

**Comment se fait la requantification dans le contexte psychoacoustique ?**

Après avoir appliqué la MDCT, et analyser les seuils de masquage via la modélisation psychoacoustique, on dispose d'une carte de masquage qui nous indique, pour chaque bande de fréquence :

● les zones inaudibles : on peut carrément mettre les coefficients à zéro (suppression) ● les zones peu audibles : on applique une quantification grossière (moins de bits),

● les zones très audibles : on applique une quantification fine (plus de bits pour éviter la dégradation sonore).

Ainsi, le nombre de bits alloués dépend de la perceptibilité du son. C’est une quantification psycho acoustiquement pondérée

1. **Algorithmes de compression :**
2. **Huffman + Masquage Psychoacoustique :**

L’approche Huffman + Masquage psychoacoustique constitue le cœur du mécanisme de compression avec perte du format «.irm ». Elle repose sur une stratégie en deux temps : une réduction perceptuelle des données basée sur la sensibilité auditive humaine, suivie d’un encodage entropique optimal. Principe : Tout d’abord, une analyse psychoacoustique est réalisée sur les coefficients MDCT pour identifier les composantes fréquentielles qui sont inaudibles ou faiblement perceptibles en présence d’autres sons dominants. Ces composantes sont soit supprimées, soit fortement quantifiées (avec perte d'information), ce qui diminue la quantité de données à encoder. Ensuite, les coefficients restants sont encodés à l’aide de l’algorithme de Huffman. Ce dernier attribue des codes binaires courts aux valeurs fréquentes, et des codes plus longs aux valeurs rares, réduisant ainsi efficacement la taille du fichier.

**Avantages :**

● Le masquage permet de conserver uniquement l’information significative pour l’auditeur, ce qui réduit drastiquement le volume de données.

● L’algorithme de Huffman exploite la redondance statistique des coefficients restants pour un encodage sans perte complémentaire.

● Cette combinaison maximise la compression tout en assurant une qualité sonore acceptable, car elle cible les limites perceptuelles de l’oreille humaine.

**Justification du choix :**

Cette approche a été choisie pour «.irm » car elle permet un contrôle fin de la qualité perçue en modulant le niveau de quantification selon les seuils d’audibilité. Elle est particulièrement efficace pour les environnements où la bande passante ou l’espace de stockage est limité, tout en préservant une excellente intelligibilité sonore.

Cette technique représente un compromis intelligent entre réduction de données et fidélité auditive, parfaitement aligné avec les objectifs du format «.irm ».

1. **LZW + Masquage psychoacoustique :**

L’algorithme LZW est une méthode de compression basée sur la détection de motifs répétitifs dans une séquence. En le combinant avec une étape de masquage psychoacoustique, on élimine d’abord les composantes inaudibles, puis on applique LZW pour exploiter les redondances restantes dans les séquences audio.

**Avantages :**

● Bonne performance sur les signaux avec structures répétitives (ex. musique électronique, sons synthétiques).

● Masquage psychoacoustique réduit la quantité de données d’entrée pour LZW, augmentant ainsi son efficacité.

**Pourquoi cette combinaison ?**

Le masquage psychoacoustique permet d’alléger le contenu à traiter en supprimant les informations inaudibles, ce qui augmente la densité des motifs utiles pour LZW. Le résultat est une compression plus agressive sans compromettre la qualité perçue. Cette combinaison est particulièrement adaptée aux signaux présentant des motifs réguliers et prévisibles

1. **Huffman + LZW + Masquage psychoacoustique :**

**Principe :**

Cette combinaison exploite les points forts de chaque méthode. Le masquage psychoacoustique élimine les données inaudibles, réduisant d’emblée la quantité d’information utile. Ensuite, LZW est appliqué pour capter les répétitions et les structures redondantes dans les coefficients restants, ce qui permet une première compression. Enfin, les symboles produits par LZW sont compressés une seconde fois via Huffman, qui optimise encore la taille des codes transmis.

**Avantage :**

Cette combinaison assure une compression très poussée tout en gardant une qualité audio élevée. Chaque étape agit sur une dimension différente de la redondance : perceptuelle (masquage), structurelle (LZW) et statistique (Huffman)

1. **Huffman seulement :**

**Principe :**

L'algorithme de Huffman utilise un codage à longueur variable basé sur la fréquence d'apparition des symboles dans le signal audio. Les échantillons ou valeurs qui apparaissent fréquemment reçoivent des codes courts, tandis que les valeurs rares sont représentées par des codes plus longs. Un arbre binaire est construit pour générer ces codes de manière optimale.

**Avantage :**

Cette méthode est relativement simple à implémenter et offre une compression sans perte. Elle est particulièrement efficace pour les signaux dont la distribution statistique est fortement asymétrique.

**Inconvénient :**

Utilisé seul, l'algorithme de Huffman ne peut pas exploiter les redondances structurelles ou séquentielles dans le signal audio, ni tenir compte des propriétés psychoacoustiques. Son taux de compression demeure donc limité comparativement aux approches combinées.

1. **LZW seulement :**

**Principe :**

L'algorithme LZW (Lempel-Ziv-Welch) fonctionne en identifiant et encodant les séquences répétitives dans le flux de données audio. Il construit dynamiquement un dictionnaire des motifs rencontrés, attribuant des codes à des séquences de plus en plus longues au fur et à mesure du traitement. Contrairement à Huffman qui exploite la fréquence des symboles isolés, LZW capture les répétitions de séquences entières.

**Avantage :**

LZW est particulièrement efficace pour les signaux présentant des motifs répétitifs ou des structures récurrentes. Il s'adapte automatiquement aux caractéristiques du signal traité sans nécessiter d'analyse préalable de la distribution statistique.

**Inconvénient :**

Son efficacité diminue considérablement pour les signaux audios très variables ou chaotiques. De plus, il ne tient pas compte des propriétés perceptuelles du son, traitant toutes les données comme également importantes, ce qui limite son taux de compression par rapport aux méthodes exploitant les spécificités de l'audition humaine.

## **Conception de l’interface graphique :**

### **Introduction :**

Après avoir exposé le contexte général du projet et abordé la problématique traitée, il est primordial de procéder à une étape de conception de qualité afin de garantir la réussite du projet. Ce chapitre se concentre sur cette étape essentielle. Nous commençons par présenter le langage de modélisation utilisé, qui permet de représenter de manière structurée les différentes composantes du projet. De plus, nous détaillons le logiciel spécifique utilisé pour faciliter la conception. Ensuite, nous mettons en évidence les acteurs qui interviennent dans le projet, en identifiant leurs rôles et leurs responsabilités respectives. Enfin, nous explorons les différents diagrammes statiques et dynamiques qui ont été élaborés pour visualiser et décrire la structure et le fonctionnement du projet de manière claire et précise.

### **Langage de modélisation UML :**

L’Unified Modeling Language (UML), qui signifie ”langage de modélisation unifié” en français, est un langage graphique utilisé pour modéliser, visualiser, spécifier et documenter les systèmes logiciels. Il permet d’appliquer les concepts de la programmation orientée objet, qui sont étroitement liés à la réalité, pour représenter de manière structurée les différentes composantes d’un système. L’UML est un outil puissant et largement utilisé dans le domaine du développement logiciel. L’UML fournit une notation standardisée composée de différents types de diagrammes qui permettent de représenter visuellement les différents aspects d’un système, tels que sa structure, son comportement, ses interactions et ses relations avec d’autres systèmes. L’utilisation du langage UML a plusieurs avantages :

• Indépendant de tout langage de programmation.

• Facilite la compréhension des représentations abstraites complexes.

• Offre un ensemble de diagrammes qui permettent de décrire et de représenter visuellement les différentes perspectives d’un système logiciel.

• Clair et précis.

### **Logiciel utilisé :**



Pour notre projet, nous avons décidé d’utiliser StarUML qui est développé par MKLabs, en raison de ses fonctionnalités avancées qui répondent à nos besoins. StarUML prend en charge tous les types de diagrammes UML nécessaires à notre projet, ce qui nous permet de représenter de manière complète et précise la structure, le comportement et les interactions du système. Cette compatibilité avec les différents types de diagrammes nous offre une flexibilité et une puissance supplémentaires pour modéliser efficacement notre solution.

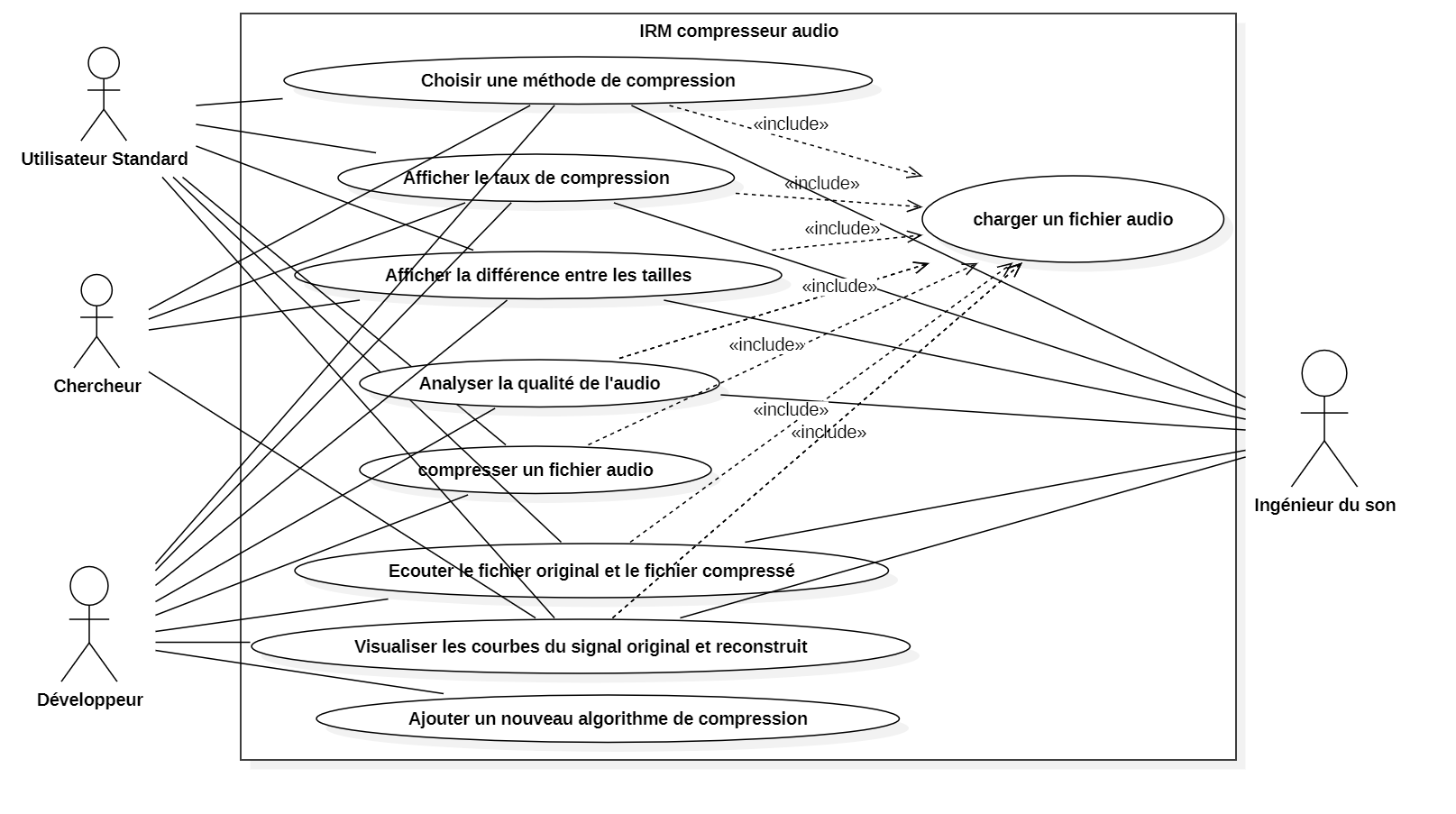
### **Identification des acteurs :**

L'acteur dans un diagramme de cas d'utilisation est toute entité qui joue un rôle dans un système donné. Il peut s'agir d'une personne, d'une organisation ou d'un système externe qui interagit avec notre application ou notre système. Pour notre compresseur audio IRM, nous identifions quatre acteurs principaux qui interagissent avec le système :

* **L'Utilisateur Standard :** il utilise les fonctionnalités de base du système comme charger un fichier audio, choisir une méthode de compression, afficher le taux de compression, comparer les tailles de fichiers et écouter les fichiers originaux et compressés.
* **Le Chercheur :** en plus des fonctionnalités de l'utilisateur standard, il peut analyser la qualité de l'audio et visualiser les courbes des signaux originaux et reconstruits.
* **Le Développeur :** il peut accéder à l'ensemble des fonctionnalités, y compris l'ajout de nouveaux algorithmes de compression.
* **L'Ingénieur du son :** il interagit principalement avec les fonctionnalités d'analyse et de comparaison de la qualité audio ainsi que le chargement et la compression des fichiers.

### **Diagramme de cas d’utilisation.**

Le diagramme de cas d’utilisation permet de modéliser le comportement d’un système et permettant de capturer ses exigences, ainsi il décrit les fonctionnalités générales et la portée d’un système. Après avoir étudié les différents cas d’utilisation que les clients peuvent réaliser via l’application, nous avons identifié un ensemble de cas d’utilisation qui sont représentés dans le diagramme de cas d’utilisation ci-dessous



Ce diagramme de cas d'utilisation représente les fonctionnalités du système de compression audio IRM et les interactions entre les différents acteurs et ces fonctionnalités.

**Présentation générale :**

Le diagramme illustre un système nommé "IRM compresseur audio" qui permet de manipuler des fichiers audios en appliquant différentes méthodes de compression. Il met en évidence quatre types d'utilisateurs (acteurs) qui interagissent avec le système, chacun ayant des niveaux d'accès et des besoins spécifiques.

**Les acteurs :**

Le système identifie quatre acteurs principaux :

* **Utilisateur Standard** : utilisateur avec des besoins basiques de compression audio
* **Chercheur** : utilisateur avec des besoins analytiques plus avancés
* **Développeur** : utilisateur technique pouvant étendre les fonctionnalités du système
* **Ingénieur du son** : spécialiste ayant des besoins spécifiques liés à la qualité audio

**Les cas d'utilisation :**

Le système propose huit fonctionnalités principales (cas d'utilisation) :

* **Charger un fichier audio** : fonctionnalité centrale permettant l'importation des fichiers à traiter
* **Choisir une méthode de compression** : sélection de l'algorithme à appliquer
* **Afficher le taux de compression** : visualisation des performances de compression
* **Afficher la différence entre les tailles** : comparaison entre fichiers originaux et compressés
* **Analyser la qualité de l'audio** : évaluation des aspects qualitatifs post-compression
* **Compresser un fichier audio** : application effective de la compression
* **Écouter le fichier original et le fichier compressé** : comparaison auditive
* **Visualiser les courbes du signal original et reconstruit** : analyse visuelle des signaux
* **Ajouter un nouvel algorithme de compression** : extension des capacités du système

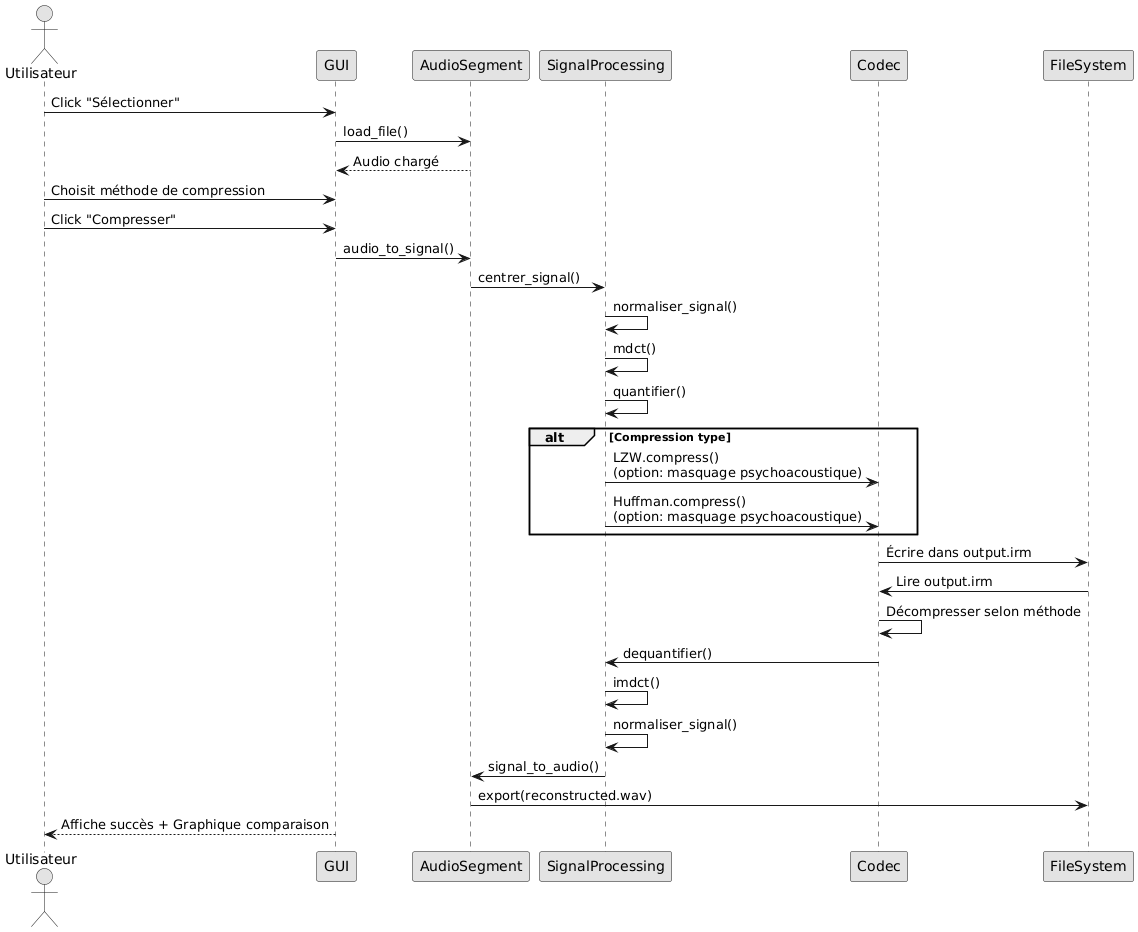
**Relations et inclusions :**

Le diagramme utilise deux types de relations :

* Les lignes pleines indiquent les fonctionnalités auxquelles chaque acteur a accès
* Les lignes pointillées avec la mention «include» montrent les dépendances entre fonctionnalités

Notamment, plusieurs fonctionnalités incluent "Charger un fichier audio", indiquant que cette opération est préalable à d'autres actions comme la compression ou l'analyse.

### **Diagramme de séquences.**



**1. Introduction au système**

Le diagramme de séquence présenté illustre un système complet de compression audio composé de cinq composants principaux :

* GUI (Interface graphique utilisateur)
* AudioSegment (Gestion des segments audio)
* SignalProcessing (Traitement du signal)
* Codec (Compression/décompression)
* FileSystem (Système de fichiers)

Ce système permet à l'utilisateur de sélectionner un fichier audio, de le compresser selon une méthode choisie, puis de le décompresser et de comparer les résultats.

**2. Description du flux d'exécution**

Le processus complet se déroule comme suit :

1. L'utilisateur commence par cliquer sur "Sélectionner" dans l'interface graphique
2. La GUI lance la fonction load\_file() vers le composant AudioSegment
3. Une fois l'audio chargé, une confirmation est renvoyée à l'utilisateur
4. L'utilisateur choisit une méthode de compression
5. L'utilisateur clique sur "Compresser" pour lancer le processus
6. Le segment audio est converti en signal via audio\_to\_signal()
7. Le signal subit plusieurs transformations :
   * Centrage (centrer\_signal())
   * Normalisation (normaliser\_signal () )
   * Application de la MDCT (mdct () )
   * Quantification (quantifier () )
8. Le codec applique ensuite la compression choisie :
   * Soit LZW
   * Soit Huffman
   * Avec option possible de masquage psychoacoustique
9. Le résultat est écrit dans un fichier "output.irm"
10. Pour la décompression, le système lit le fichier "output.irm"
11. Le processus inverse est appliqué :
    * Déquantification (dequantifier ())
    * MDCT inverse (imdct ())
    * Normalisation du signal (normaliser\_signal ())
    * Conversion du signal en audio (signal\_to\_audio ())
12. Le fichier audio reconstruit est exporté au format WAV
13. L'utilisateur reçoit une confirmation de succès et un graphique comparant l'original et le résultat

**3. Architecture du système**

L'architecture présente une séparation claire des responsabilités :

* **GUI** : Gère toutes les interactions avec l'utilisateur et coordonne le flux de travail
* **AudioSegment** : Responsable du chargement et de la gestion des données audio brutes
* **SignalProcessing** : Effectue toutes les transformations mathématiques sur le signal
* **Codec** : Applique les algorithmes de compression et décompression
* **FileSystem** : Gère l'écriture et la lecture des fichiers

Cette séparation permet une modularité du système, où chaque composant peut être modifié indépendamment sans affecter les autres, tant que les interfaces restent compatibles.

### **Conclusion :**

Au cours de ce chapitre, nous avons exploré en détail les différents diagrammes qui décrivent les fonctionnalités de notre interface graphique. Cette analyse approfondie nous permettra de prendre des décisions éclairées concernant les méthodes et les outils à utiliser lors de la réalisation de notre projet. En comprenant clairement les besoins et les exigences de notre application grâce à ces diagrammes, nous serons en mesure de choisir les approches les plus adaptées pour garantir le succès de notre projet.

## **Outils Techniques :**

Cette interface graphique pour un compresseur audio utilise un ensemble complet d'outils et de technologies pour créer une application de bureau fonctionnelle. Voici une analyse détaillée des principaux outils techniques employés :

**1. PySide6 (Qt pour Python) :**

La bibliothèque principale utilisée pour construire l'interface graphique est PySide6, qui est l'implémentation officielle de Qt pour Python développée par The Qt Company.

* **QApplication** : Point d'entrée de l'application Qt
* **QWidget** : Widget de base servant de conteneur principal
* **QVBoxLayout/QHBoxLayout/QGridLayout** : Systèmes de mise en page pour organiser les éléments
* **QPushButton** : Boutons interactifs pour déclencher des actions
* **QFileDialog** : Boîte de dialogue pour sélectionner des fichiers
* **QLabel** : Affichage de texte et d'informations
* **QComboBox** : Liste déroulante pour sélectionner la méthode de compression
* **QSplitter** : Division de l'interface en panneaux redimensionnables
* **QGroupBox** : Regroupement d'éléments liés dans des boîtes avec titres
* **QFrame** : Cadres pour organiser visuellement le contenu

Le logo de Qt pour python est :



**2. Matplotlib :**

Utilisé pour la visualisation des signaux audio :

* **FigureCanvasQTAgg** : Intégration de graphiques Matplotlib dans des applications Qt
* **plt.subplots** : Création de sous-graphiques pour comparer les signaux originaux et reconstruits

Son logo est :



**3. Traitement audio et signal :**

Pour le traitement du signal audio et la compression :

* **pydub (AudioSegment)** : Manipulation des fichiers audio, lecture et écriture
* **NumPy** : Traitement numérique, manipulation des tableaux et calculs mathématiques.son logo est :



* **SciPy** : Fonctions de traitement du signal, notamment :
  + **butter, filtfilt** : Filtres de signal
  + **dct, idct** : Transformée en cosinus discrète et son inverse, fondamentale pour la compression

Son logo est :



**4. Algorithmes de compression :**

L'application intègre plusieurs algorithmes de compression :

* **Huffman** : Compression par codage de Huffman via une classe personnalisée HuffmanCoder
* **LZW** : Compression Lempel-Ziv-Welch via une classe personnalisée LZWCoder
* **Modèle psychoacoustique** : Module personnalisé pour l'élimination des informations audio imperceptibles, basé sur les propriétés de perception humaine

**5. Interface utilisateur et experience:**

* **Feuilles de style CSS** : Application extensive de styles personnalisés pour une interface cohérente et moderne (thème sombre)
* **Notifications d'état** : Mise à jour dynamique des labels pour informer l'utilisateur
* **Gestion des événements** : Connexion des boutons aux fonctions appropriées
* **Structure organisée** : Division logique en sections (fichier, options, lecture, statistiques)

Cette application représente une intégration complexe de traitement du signal, d'algorithmes de compression et d'interface graphique moderne, avec une attention particulière portée à l'expérience utilisateur et au design visuel

## **Réalisation de l’interface graphique :**

### **Introduction :**

L'interface graphique du format IRM a été conçue pour offrir aux utilisateurs une expérience intuitive et fonctionnelle, tout en reflétant la nature modulaire et adaptative du format de compression audio développé. Cette interface constitue le point d'entrée principal permettant d'accéder aux différentes fonctionnalités du système, notamment la sélection des six modes de compression, la visualisation des résultats et l'analyse comparative des performances.

La conception de l'interface a été guidée par plusieurs objectifs fondamentaux :

* Permettre une utilisation intuitive des différentes stratégies de compression
* Offrir une visualisation claire des données audio avant et après compression
* Fournir des retours d'information pertinents sur le processus de compression
* Représenter visuellement les compromis entre qualité audio et taux de compression
* Assurer une expérience utilisateur cohérente malgré la complexité sous-jacente des algorithmes

L'implémentation a nécessité un équilibre entre richesse fonctionnelle et simplicité d'utilisation, afin de rendre accessibles les concepts techniques complexes du format IRM à des utilisateurs de différents niveaux d'expertise.

### **Nom et logo de l’interface :**



Le logo de notre interface graphique IRM a été conçu pour établir une identité visuelle forte et distincte, reflétant la nature innovante du format de compression audio développé.

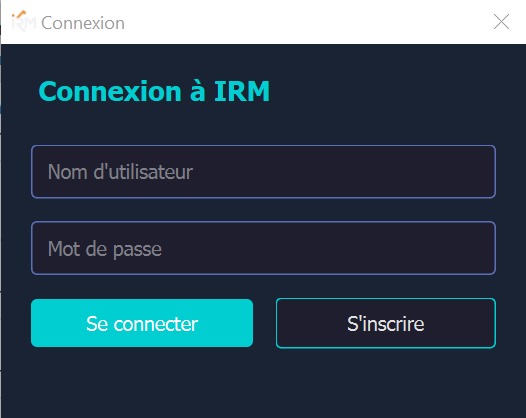
Le design adopte une approche minimaliste mais efficace, mettant en avant l'acronyme "IRM" dans une typographie moderne et robuste. La couleur bleu marine dominante évoque la fiabilité et le professionnalisme, tandis que l'élément orange dynamique apporte contraste et énergie, symbolisant l'innovation technologique au cœur du projet.

### **Interface de connexion et inscription :**

L'application débute par une interface d'authentification permettant aux utilisateurs de s'identifier ou de créer un compte. Cette interface comprend :

* Un formulaire de connexion avec :
  + Champ pour le nom d'utilisateur
  + Champ pour le mot de passe
  + Bouton "Se connecter" pour valider les informations
  + Bouton "S'inscrire" pour accéder au formulaire d'inscription
* Un formulaire d'inscription comprenant :
  + Champ pour le nom d'utilisateur
  + Champ pour le mot de passe
  + Champ pour confirmer le mot de passe
  + Sélecteur de profession (Utilisateur Standard, Chercheur, Ingénieur du son, Développeur)
  + Bouton "S'inscrire" pour créer le compte
  + Bouton "Annuler" pour revenir à l'écran de connexion

### 



### **Interface de compression pour Chercheur, Ingénieur du son et Développeur :**

L'interface principale se compose de deux onglets principaux : "Compression" et "Stats".

1. **Onglet "Compression" :**

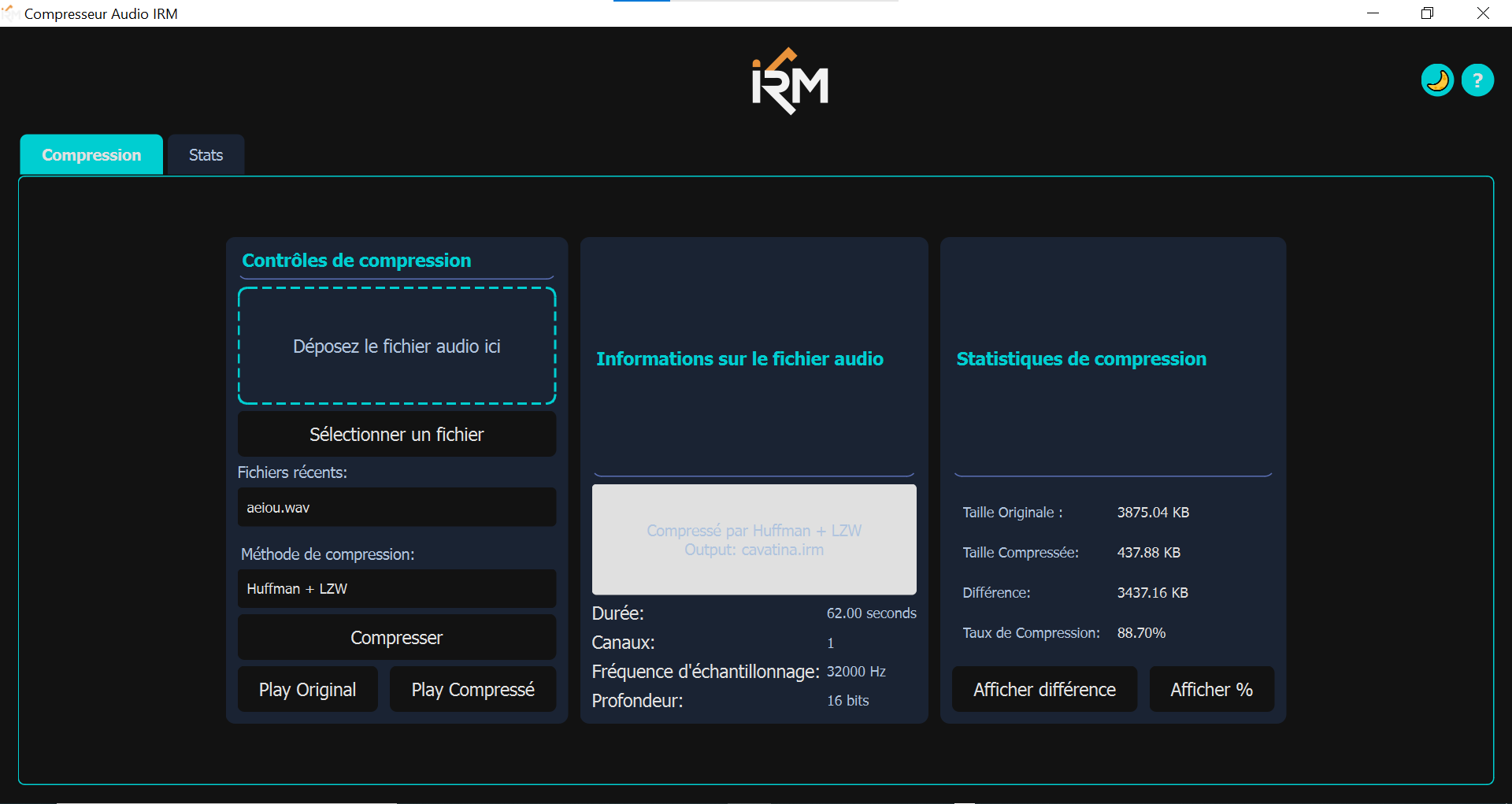
Cet onglet présente l'interface de travail principale avec :

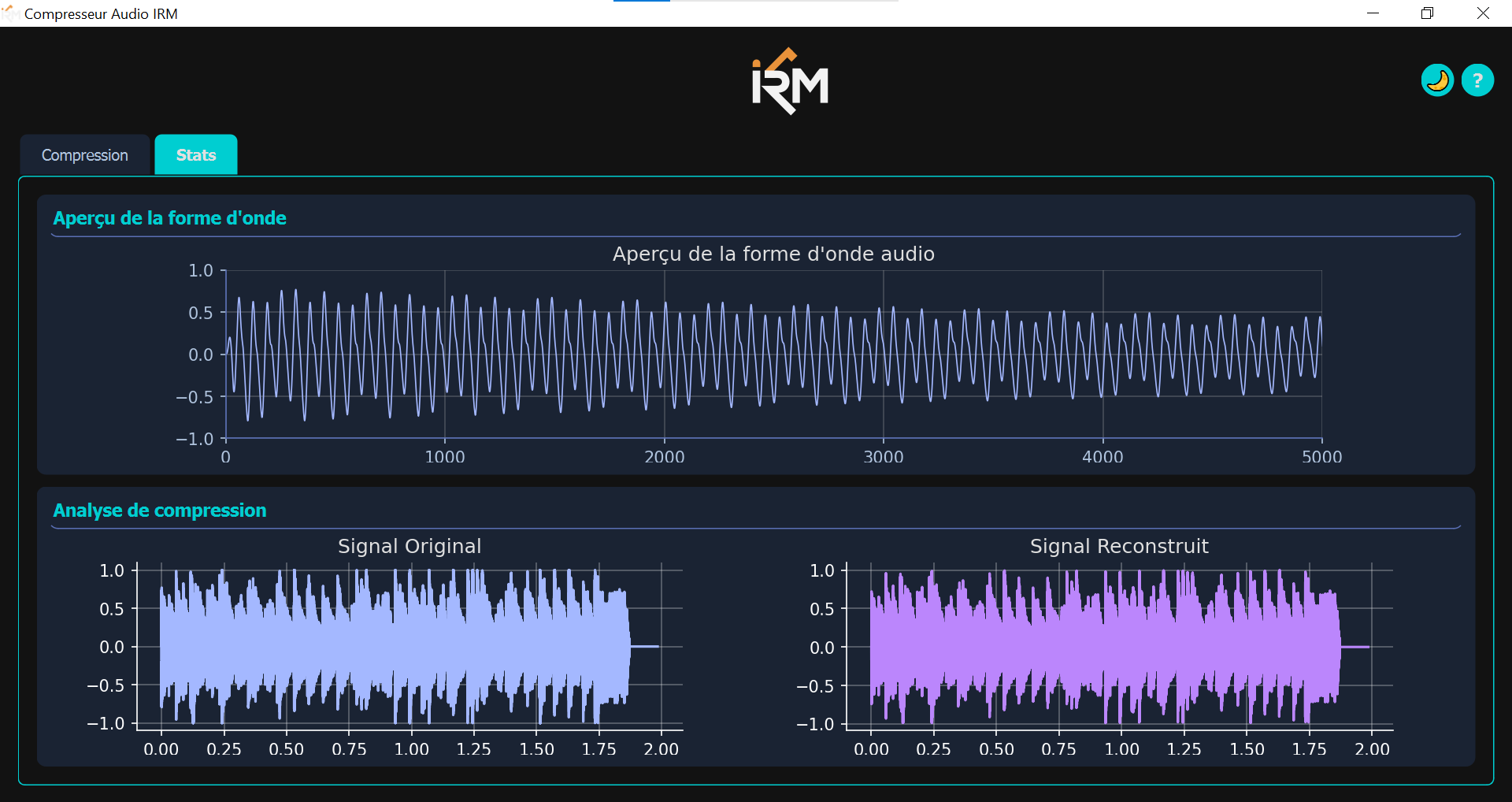
* Un panneau central "Contrôles de compression" contenant :
  + Une zone de dépôt pour glisser-déposer des fichiers audios
  + Un bouton "Sélectionner un fichier" pour parcourir l'explorateur
  + Une liste des fichiers récemment utilisés
  + Un menu déroulant pour sélectionner la méthode de compression (LZW only, Huffman + LZW)
  + Un bouton "Compresser" pour lancer le traitement
  + Un bouton "Ajouter Algorithme" (visible uniquement pour les utilisateurs avec profil Développeur)
  + Deux boutons de lecture : "Play Original" et "Play Compressé"
* Un panneau d'informations sur le fichier audio affichant :
  + Le statut de compression (méthode utilisée et nom du fichier de sortie)
  + La durée du fichier en secondes
  + Le nombre de canaux audio
  + La fréquence d'échantillonnage en Hz
  + La profondeur en bits
* Un panneau de statistiques de compression présentant :
  + La taille originale du fichier en KB
  + La taille compressée en KB
  + La différence de taille en KB
  + Le taux de compression en pourcentage
  + Boutons "Afficher différence" et "Afficher %" pour basculer entre les modes d'affichage

1. **Onglet "Stats" :**

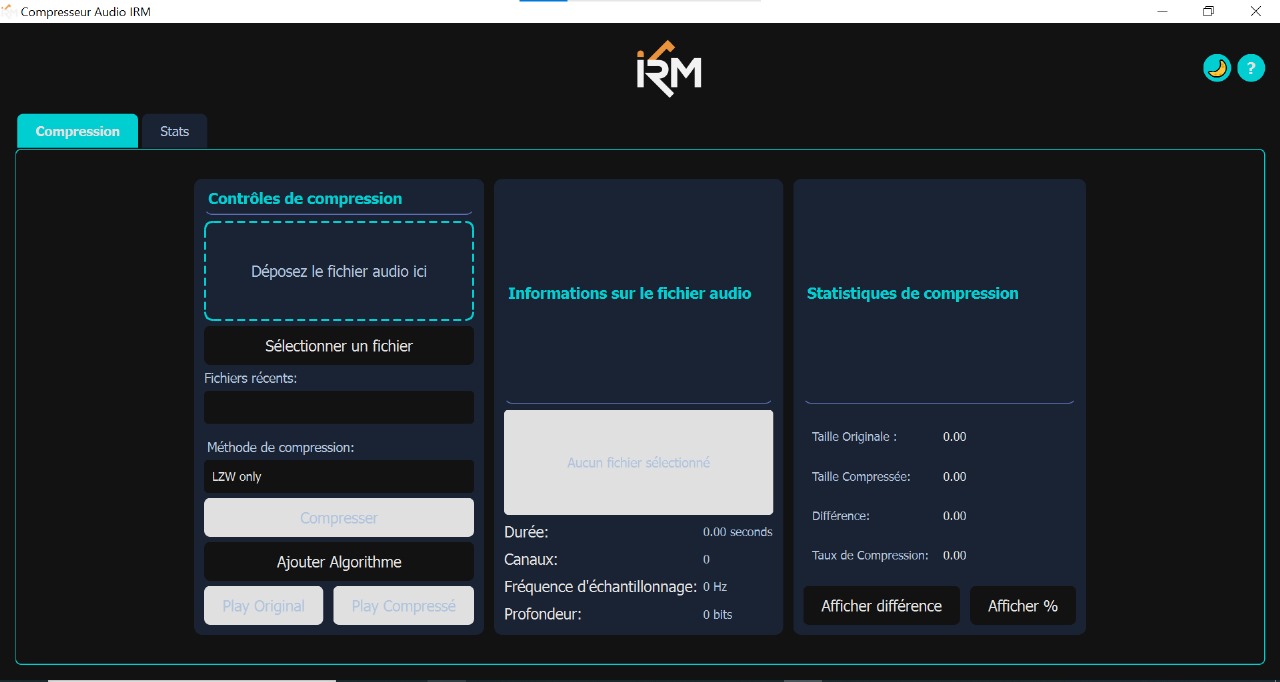
Cet onglet permet de visualiser les données audio et les résultats de compression :

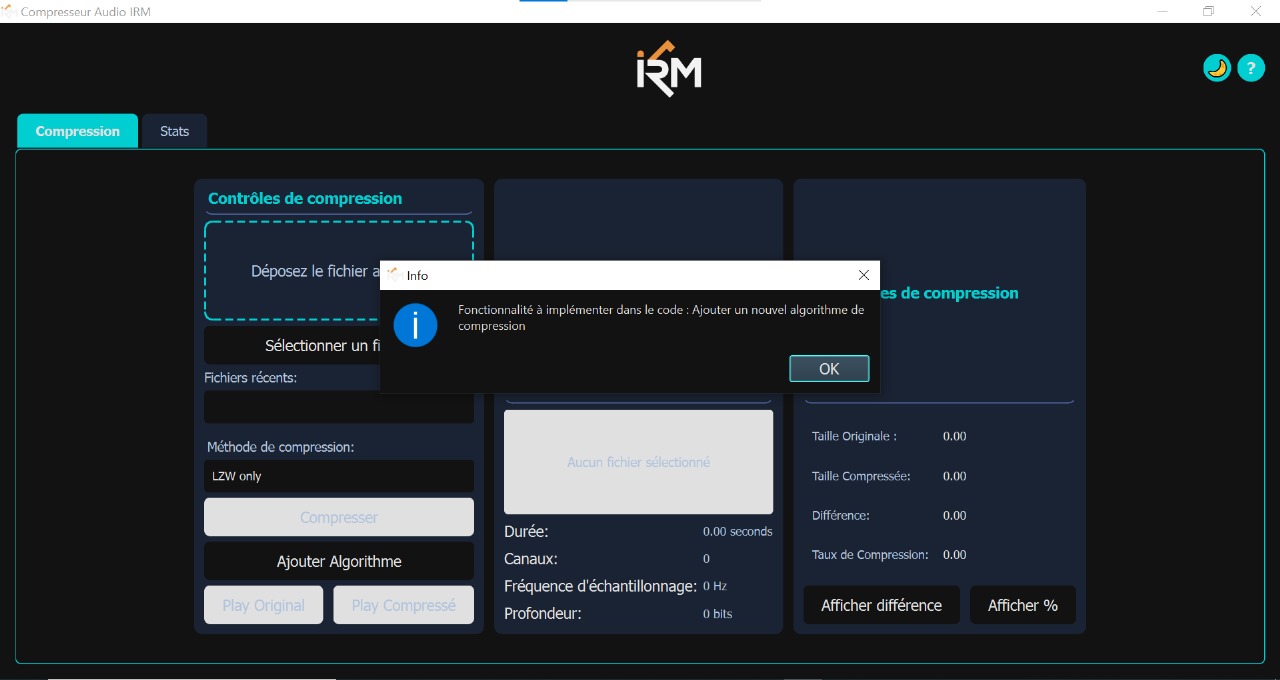
* Un graphique "Aperçu de la forme d'onde" montrant la représentation visuelle des données audio
* Une section "Analyse de compression" présentant deux graphiques comparatifs :
  + "Signal Original" : visualisation de l'onde sonore avant compression
  + "Signal Reconstruit" : visualisation de l'onde sonore après compression/décompression





Si la profession de l’utilisateur est « Développeur » cette interface serait affichée avec le bouton « Ajouter algorithme »



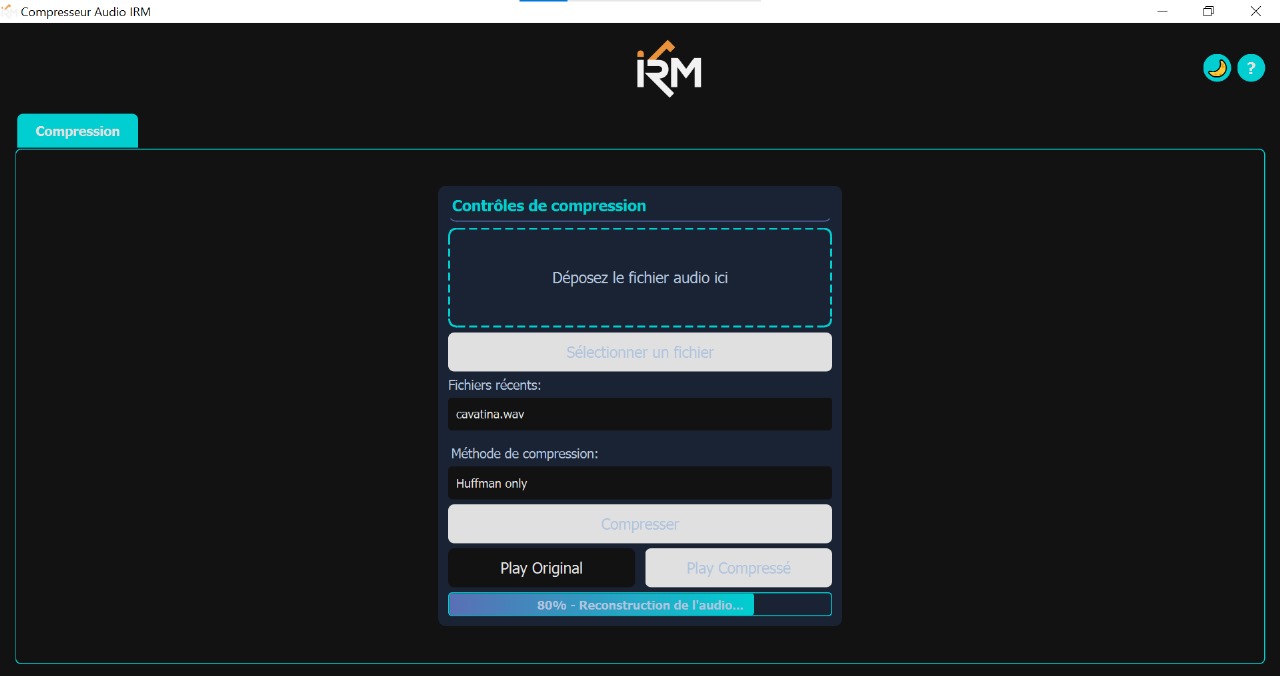


### **Interface principale de compression pour l’utilisateur standard :**

L'interface du logiciel "Compresseur Audio IRM" présente une conception épurée avec un fond sombre et des éléments d'interface en turquoise. La fenêtre principale est dominée par l'onglet "Compression" actuellement actif, visible dans la partie supérieure gauche.

Au centre de l'écran, on distingue le logo "IRM" en haut, suivi par le panneau principal "Contrôles de compression" qui occupe la majeure partie de l'interface. Ce panneau comprend :

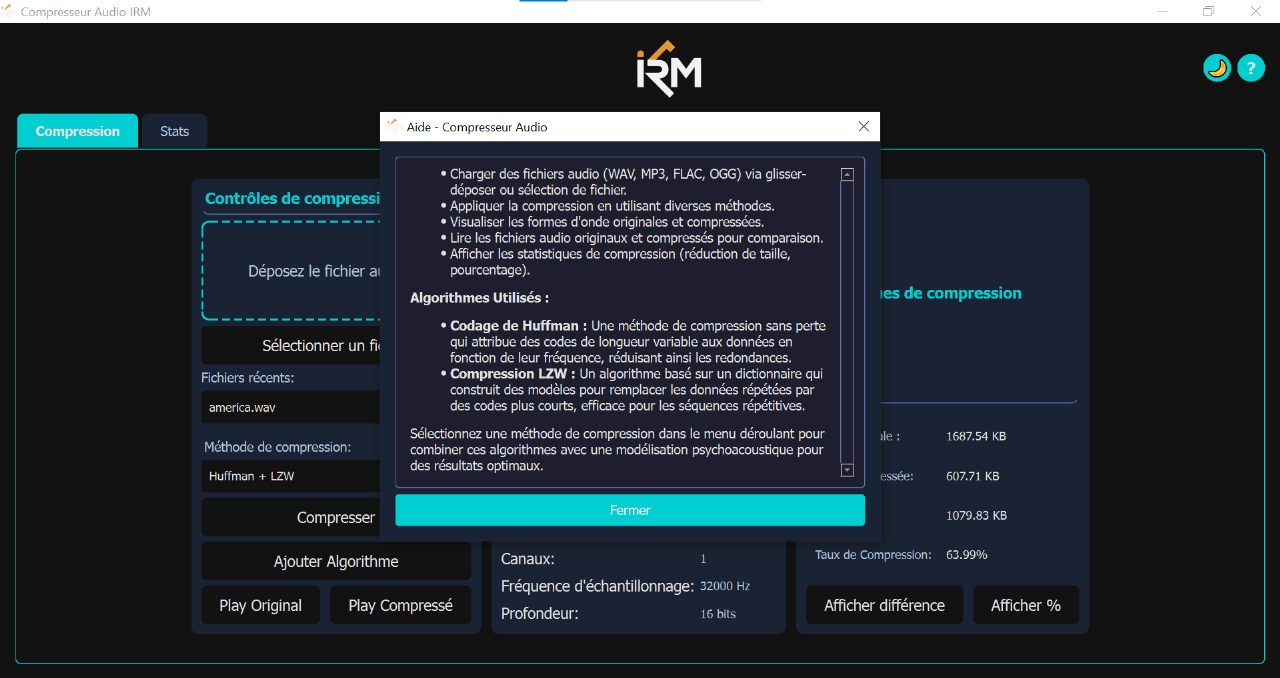
* Une zone rectangulaire délimitée par des pointillés turquoise avec l'instruction "Déposez le fichier audio ici"
* Un bouton "Sélectionner un fichier" juste en dessous
* Une section "Fichiers récents" qui affiche Une liste des fichiers récemment utilisés
* Un paramètre "Méthode de compression"
* Un bouton "Compresser" pour lancer le traitement
* Deux boutons de lecture côte à côte : "Play Original" à gauche et "Play Compressé" à droite
* Une barre de progression en bas du panneau qui indique la progression du traitement

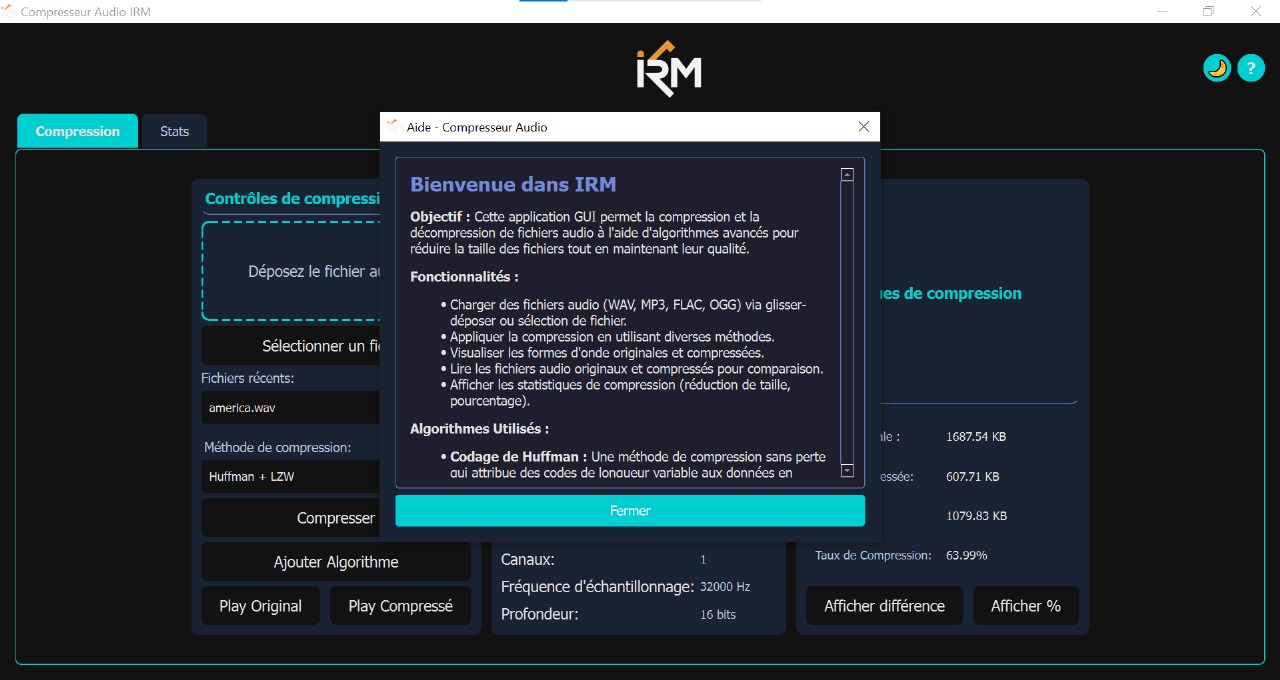


### **Système d'aide contextuelle :**

L'interface intègre un système d'aide accessible via un bouton d'information qui affiche :

* Une fenêtre "Aide - Compresseur Audio" contenant :
  + Une présentation des objectifs de l'application
  + Une liste des fonctionnalités principales
  + Une description détaillée des algorithmes utilisés (Huffman et LZW)
  + Des conseils d'utilisation pour optimiser les résultats





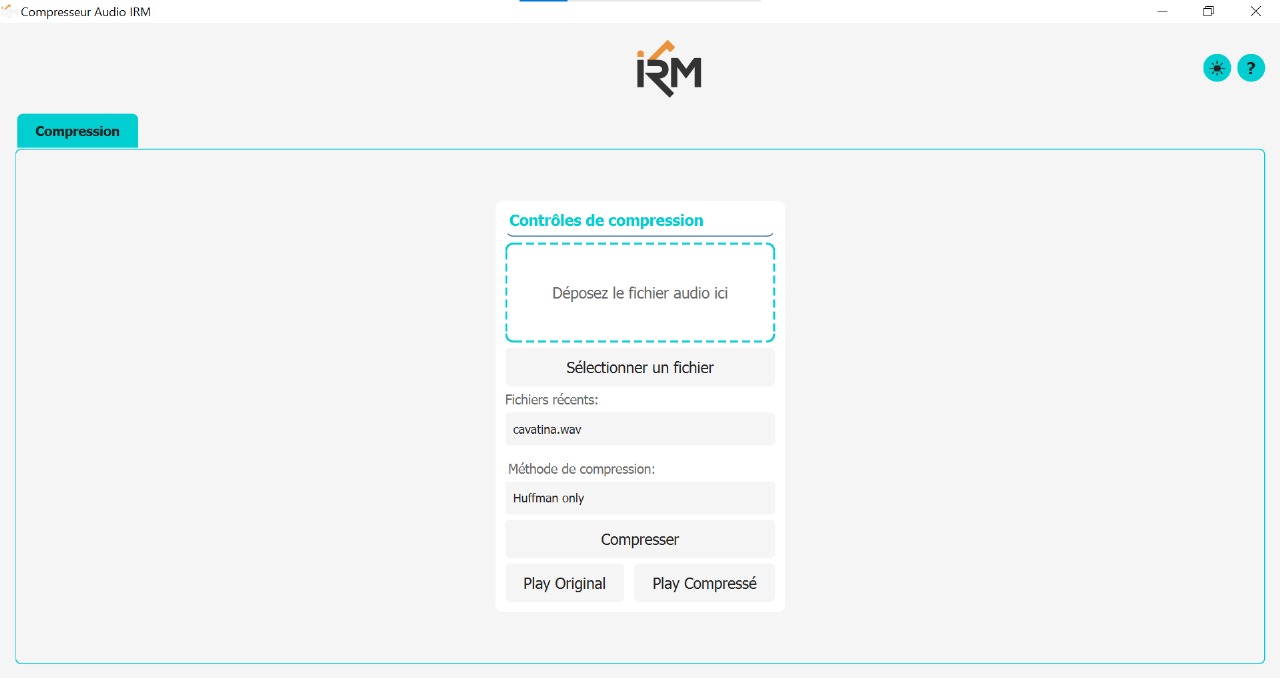
### **Personnalisation de l'expérience visuelle :**

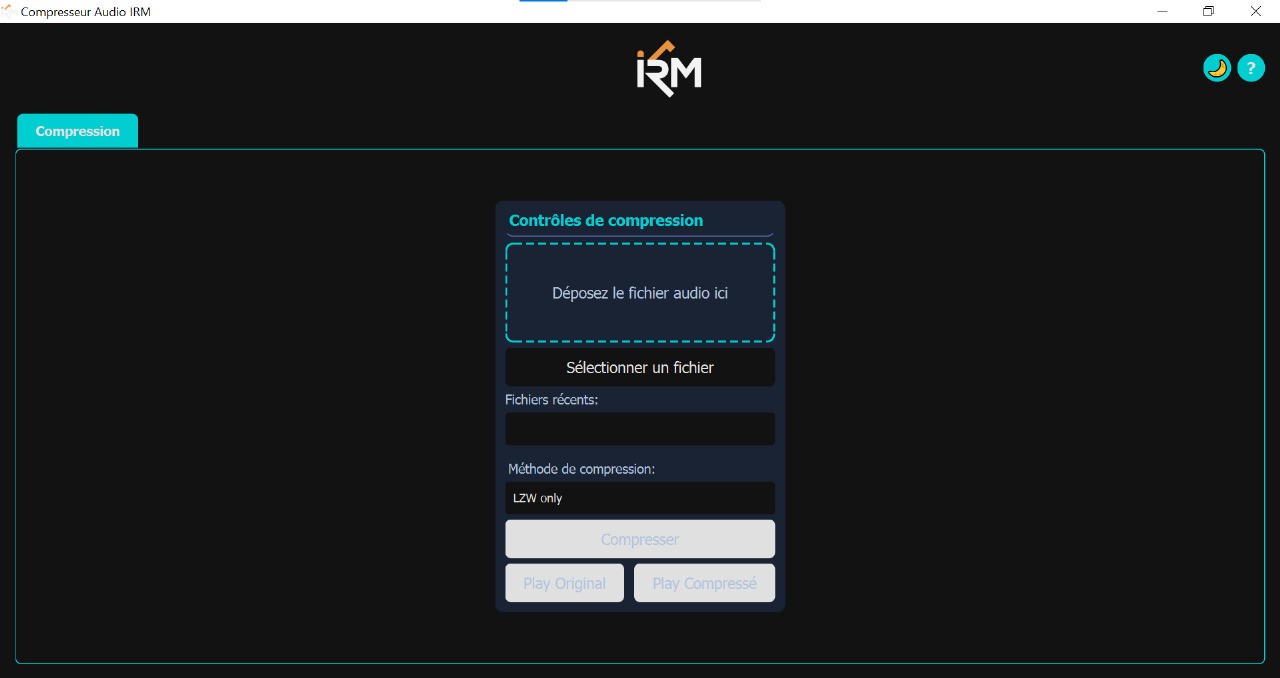
Notre application offre une flexibilité supplémentaire pour répondre aux préférences visuelles des utilisateurs :

* Option de changement de thème intégrée :
  + Mode sombre (Dark Mode) : interface à dominante sombre avec accents de couleur turquoise, optimisée pour le travail en environnement peu éclairé et pour réduire la fatigue oculaire pendant les longues sessions de travail audio
  + Mode clair (Light Mode) : interface à fond clair avec contraste optimisé, idéale pour les environnements très lumineux
* Avantages de cette fonctionnalité :
  + Amélioration du confort visuel selon les conditions d'éclairage
  + Réduction de la consommation de batterie sur certains appareils en mode sombre
  + Accessibilité accrue pour les utilisateurs ayant des préférences visuelles spécifiques
  + Personnalisation de l'expérience utilisateur
* Persistance des préférences:
  + Le choix du mode d'affichage est sauvegardé dans les préférences utilisateur
  + La sélection est conservée entre les sessions

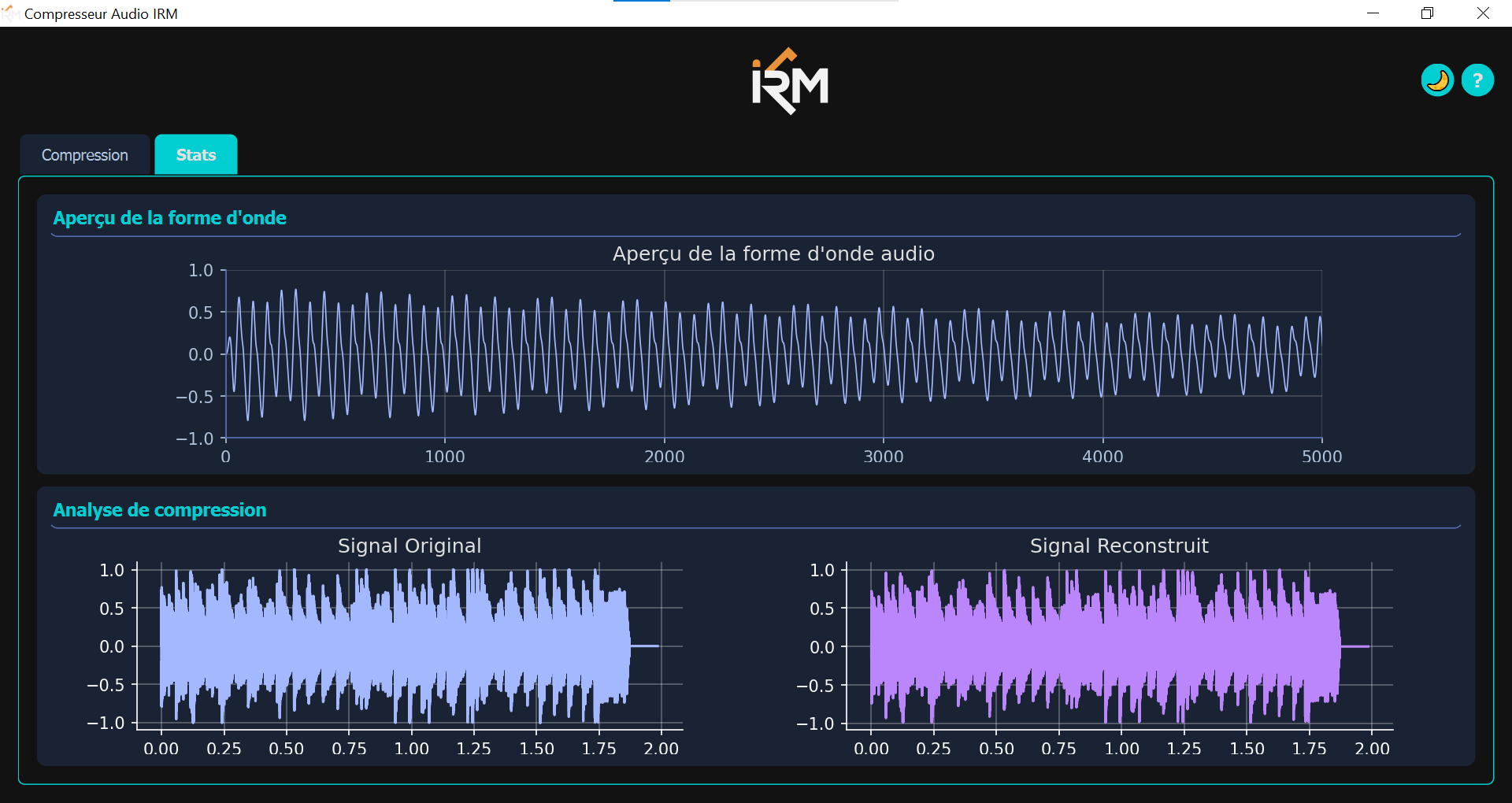
Cette fonctionnalité de personnalisation visuelle s'inscrit dans notre volonté de créer une expérience utilisateur adaptable et confortable, particulièrement importante pour une application dédiée au traitement audio qui peut nécessiter des sessions de travail prolongées.

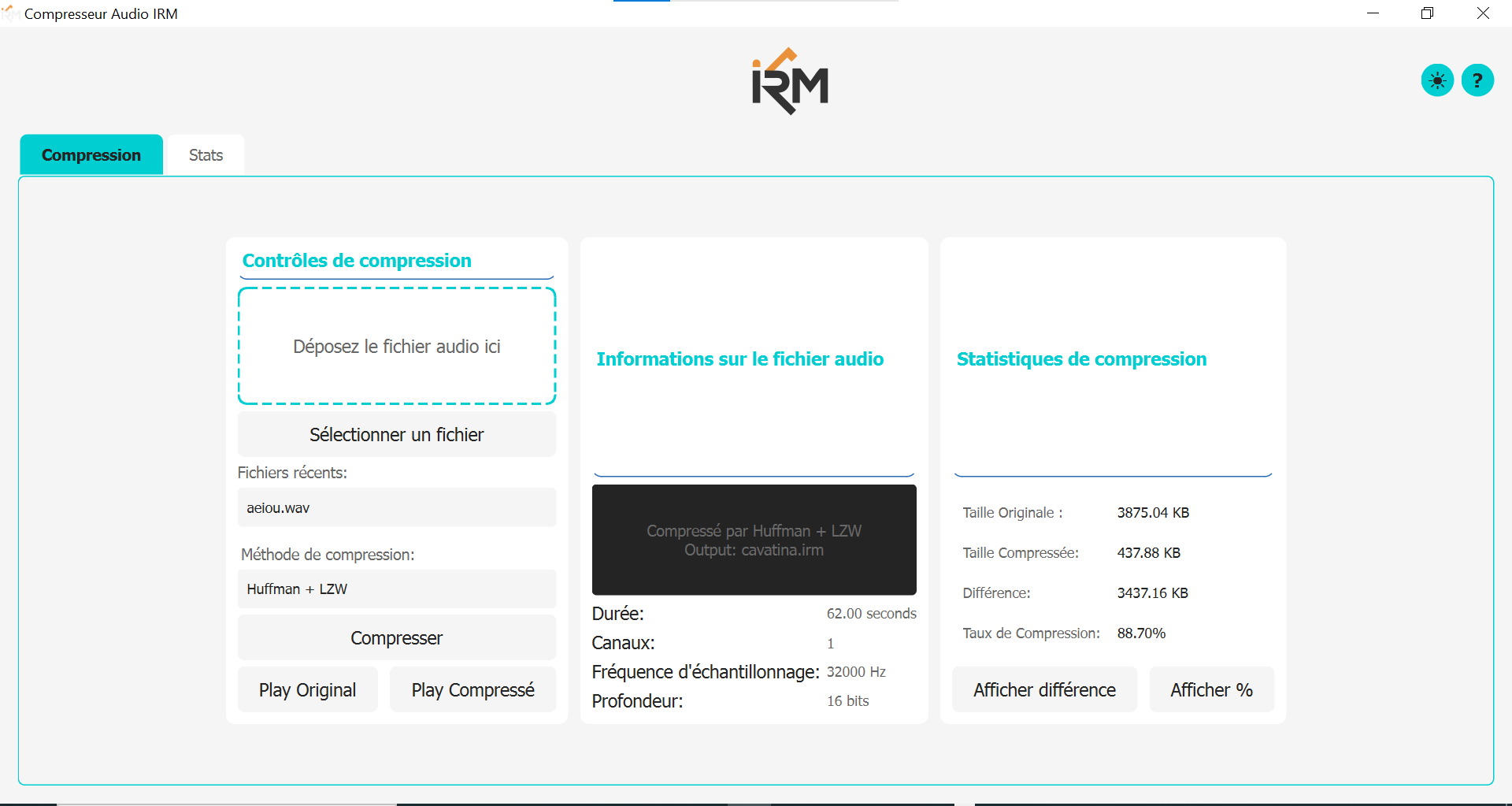
Voici quelques exemples :

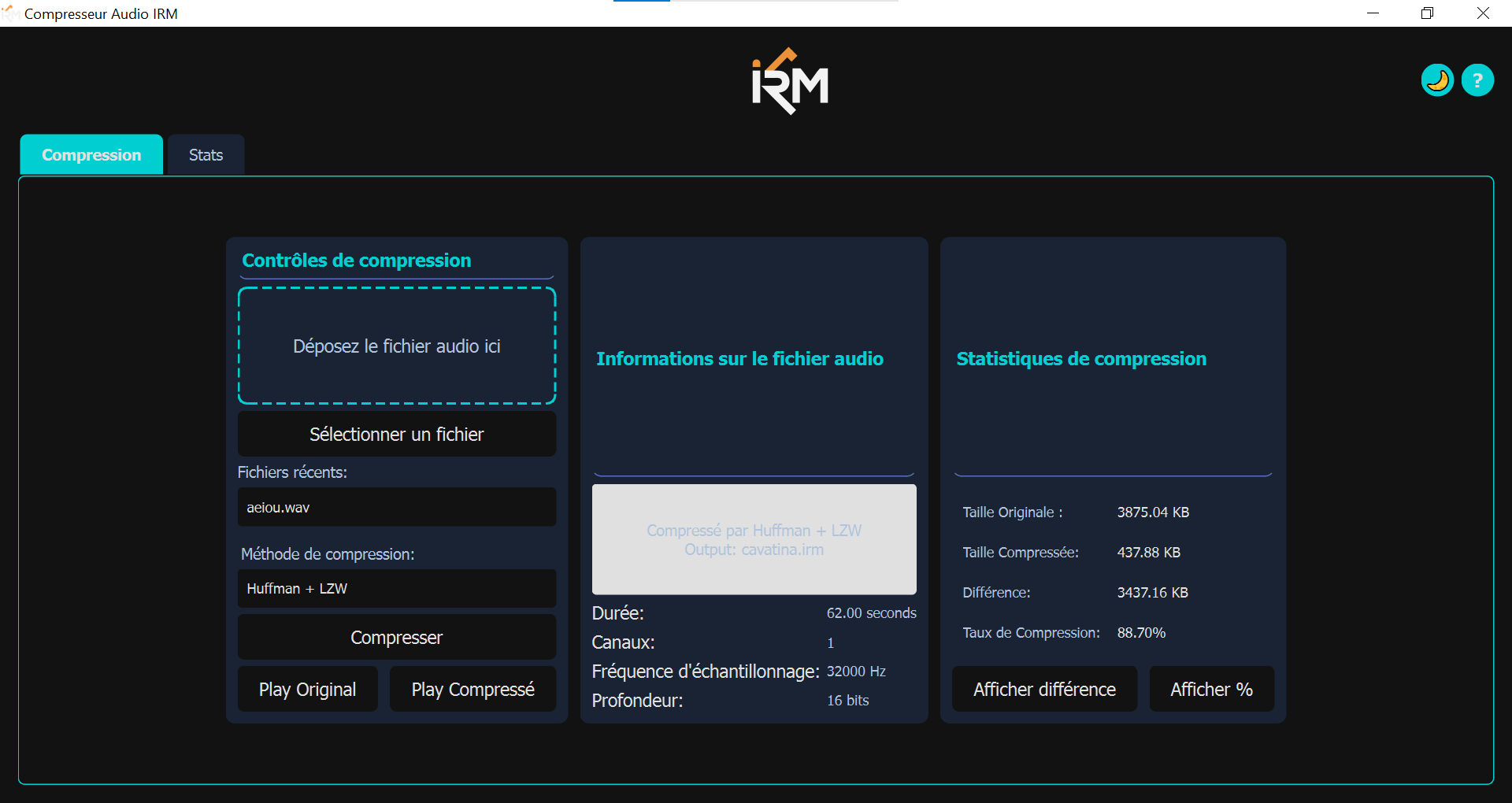












### **Innovation : Plateforme web de distribution :**

Dans le cadre de ce projet, nous avons développé une innovation majeure pour faciliter l'accès à notre application :

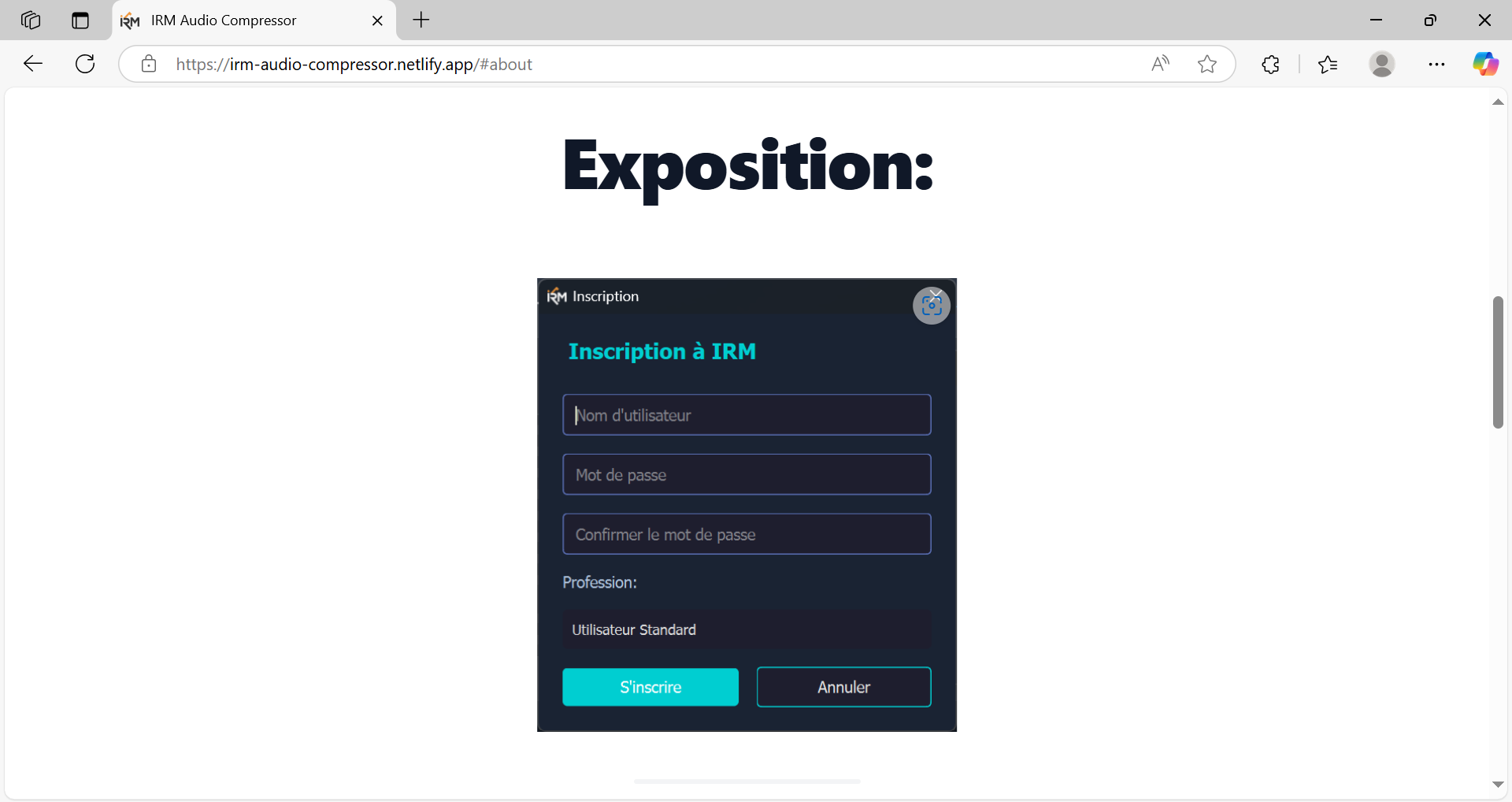
* Création d'un site web dédié permettant aux utilisateurs de :
  + Télécharger directement l'application Compresseur Audio IRM
  + Accéder à la documentation complète
  + S'inscrire en ligne pour synchroniser leur profil
  + Travailler avec l'application sans restrictions d'installation
* Cette approche présente plusieurs avantages :
  + Accessibilité accrue pour tous les types d'utilisateurs
  + Processus de mise à jour simplifié
  + Possibilité de sauvegarder les projets en ligne
* La plateforme web s'intègre parfaitement avec l'application desktop :
  + Synchronisation du compte utilisateur
  + Sauvegarde des préférences
  + Accès aux dernières mises à jour des algorithmes

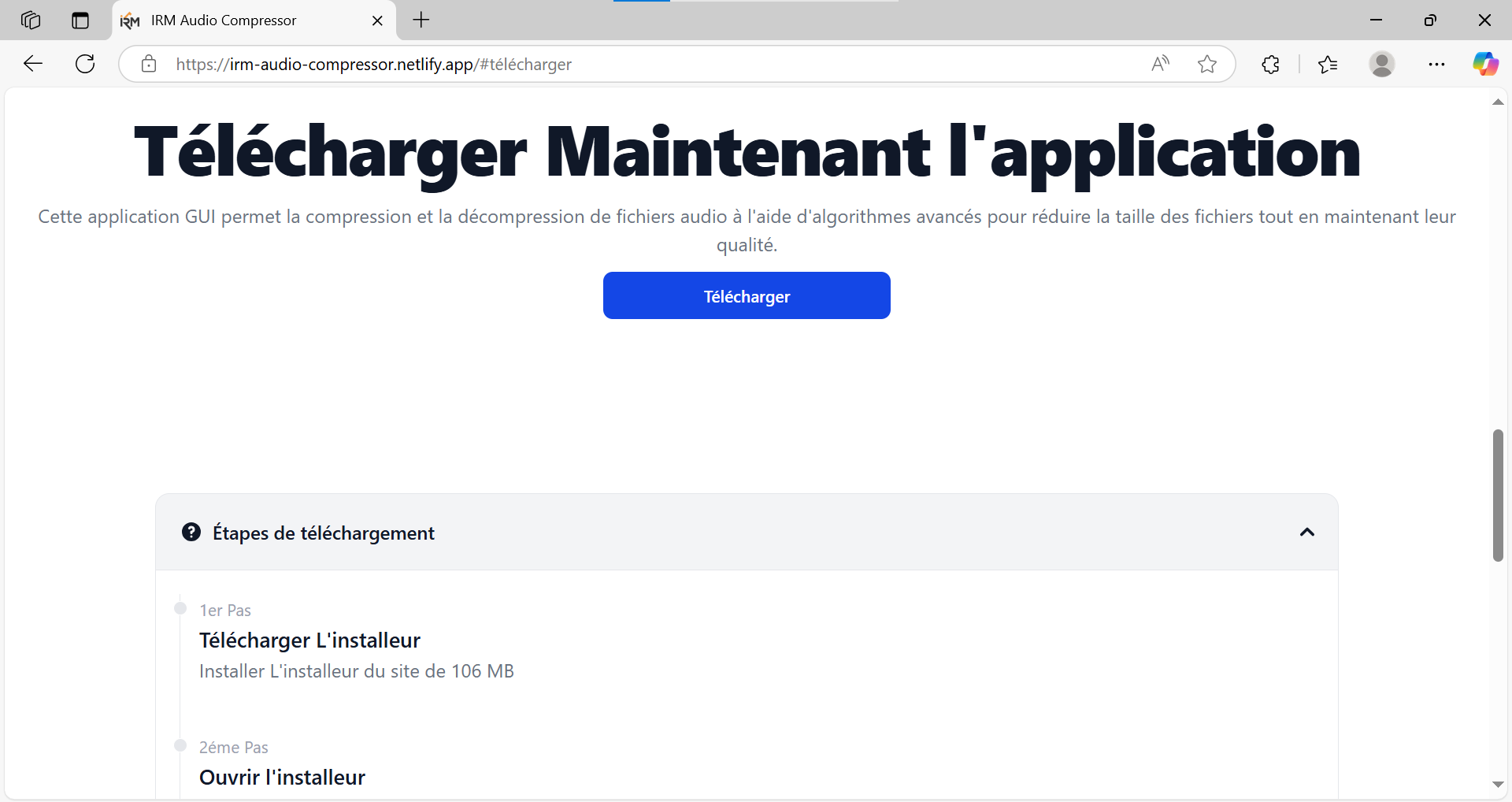
Cette innovation distingue notre solution des compresseurs audio traditionnels en offrant une expérience utilisateur complète, depuis la découverte de l'outil jusqu'à son utilisation quotidienne, tout en facilitant les mises à jour et l'intégration de nouvelles fonctionnalités.

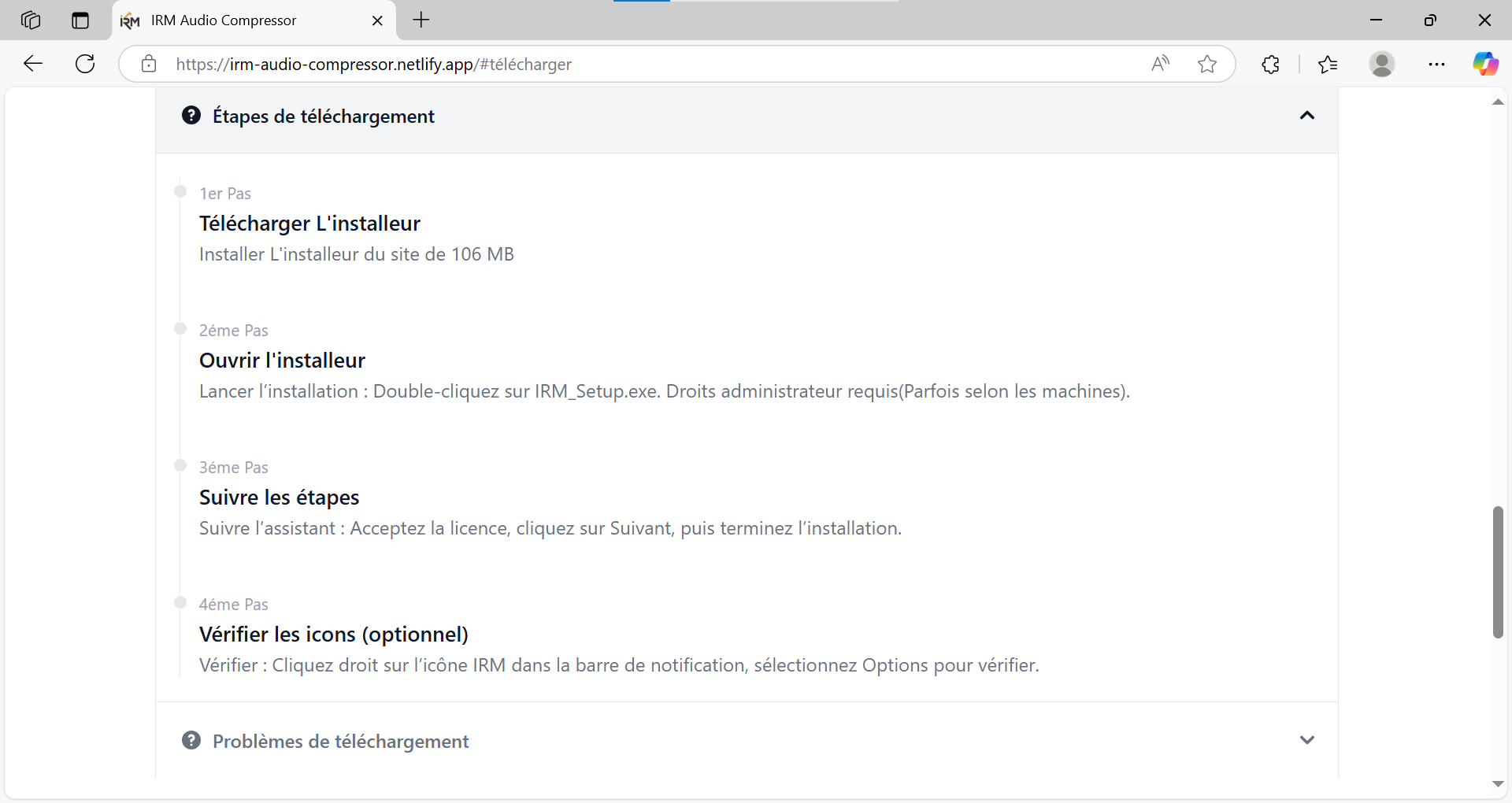
**Notre site web est accessible à l'adresse suivante :** [**IRM Audio Compressor**](https://irm-audio-compressor.netlify.app/)

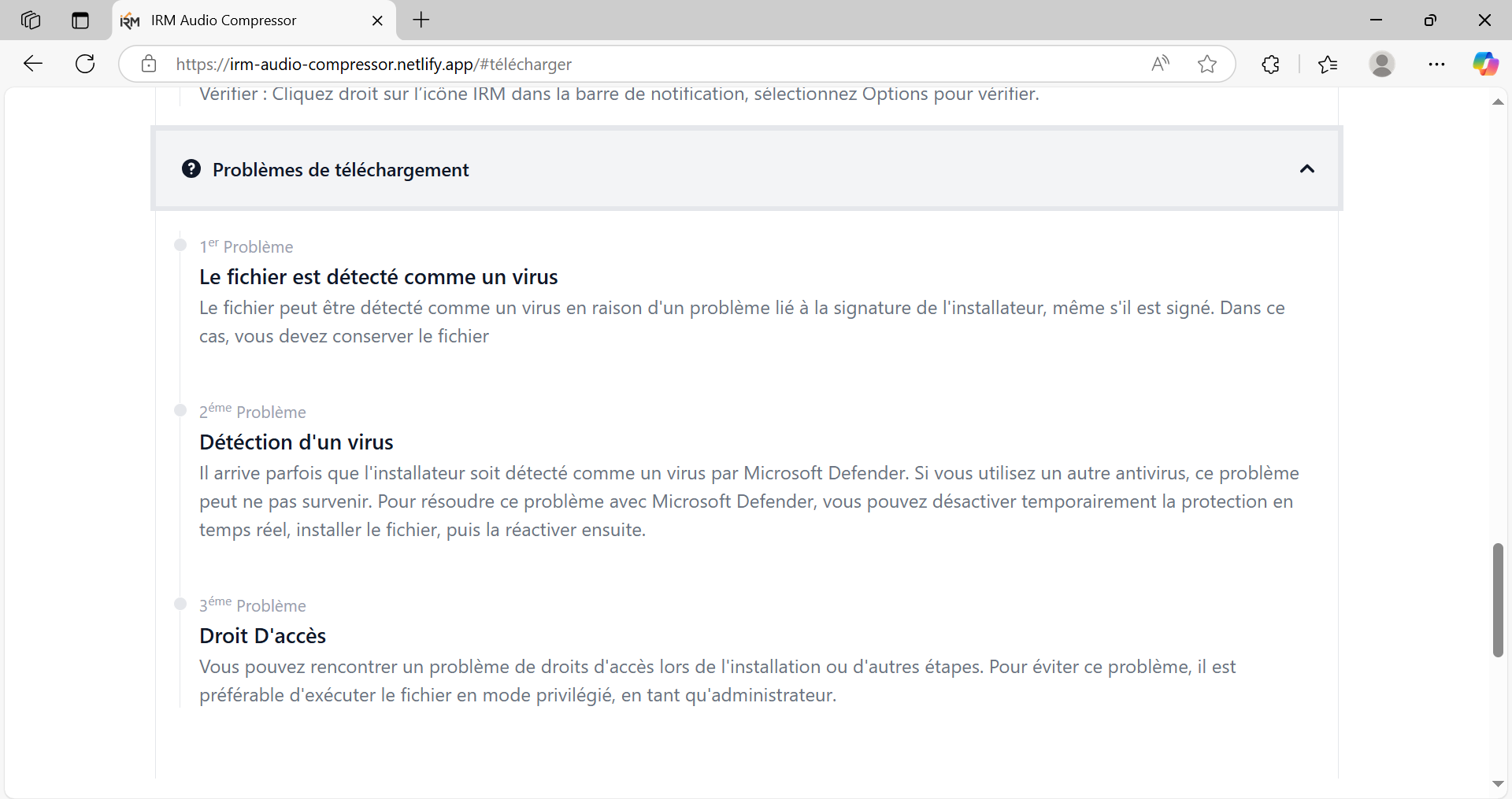
**Ou en utilisant le lien suivant : https://irm-audio-compressor.netlify.app/**

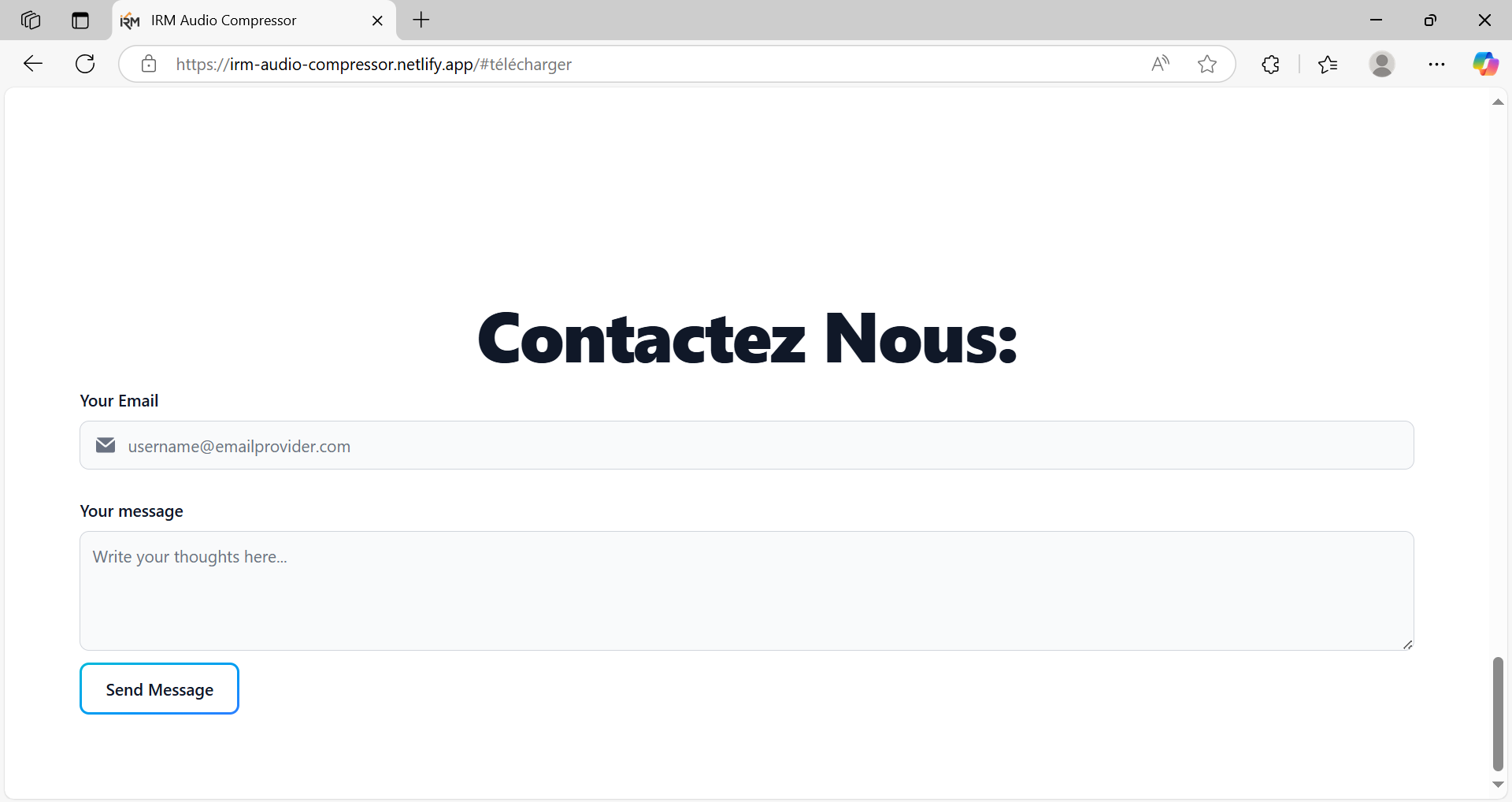












## **Résultats et évaluation**

### **Performances du format IRM :**

Le format IRM propose plusieurs modes de compression adaptés à différents besoins, ce qui lui confère une grande flexibilité. Voici une analyse détaillée des performances selon les différentes stratégies de compression implémentées :

1. **Stratégies de compression**

Le format IRM implémente six stratégies de compression distinctes, chacune offrant un compromis différent entre qualité audio, taux de compression et complexité computationnelle :

* **LZW seulement (sans perte)** : Utilise uniquement la compression LZW sur les coefficients MDCT quantifiés, offrant une compression sans perte sur les données déjà quantifiées.
* **Huffman seulement (sans perte)** : Applique uniquement le codage de Huffman, particulièrement efficace pour les distributions de coefficients non uniformes après quantification.
* **Huffman + LZW (sans perte)** : Combine les avantages des deux approches en appliquant d'abord Huffman pour exploiter les distributions statistiques, puis LZW pour capturer les redondances restantes.
* **Masquage psychoacoustique + Huffman + LZW (avec perte)** : Ajoute un modèle psychoacoustique qui alloue les bits de manière perceptuelle avant d'appliquer la double compression Huffman puis LZW.
* **Masquage psychoacoustique + Huffman (avec perte)** : Utilise le modèle psychoacoustique suivi du codage de Huffman uniquement.
* **Masquage psychoacoustique +LZW (avec perte)** : Combine le masquage perceptuel avec la compression LZW.

1. **Facteurs influençant les performances :**

Les performances du format IRM dépendent de plusieurs facteurs clés :

* 1. **Quantification**

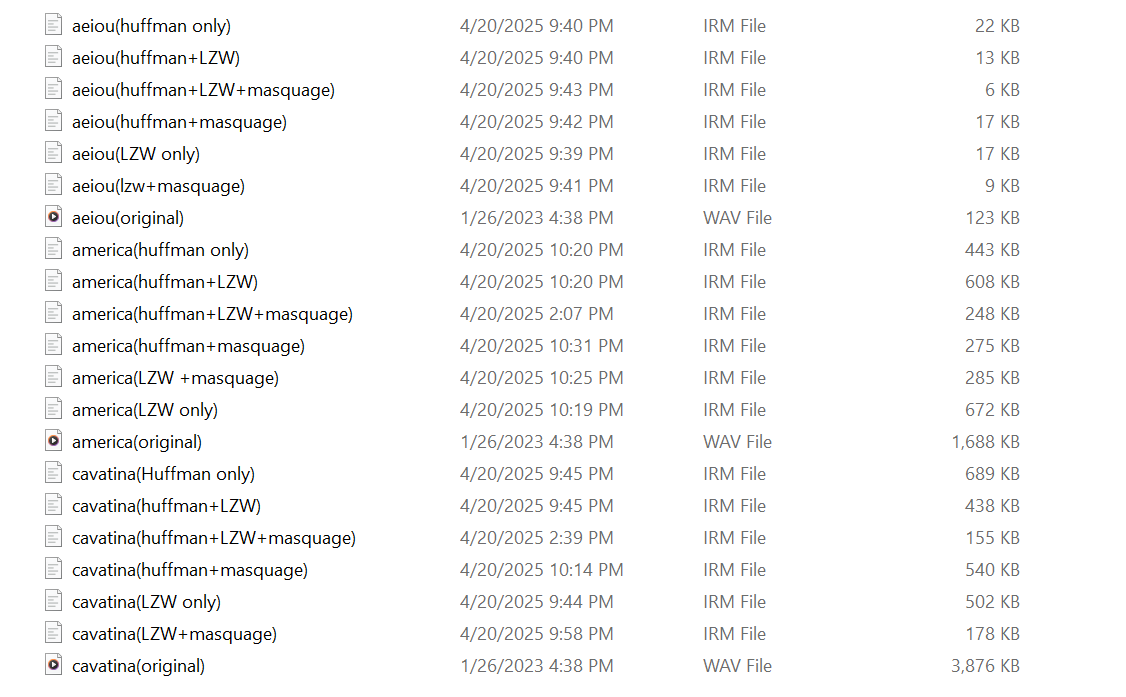
La quantification (paramètre q) constitue le premier niveau de compression avec perte. Un facteur de quantification plus élevé augmente le taux de compression mais diminue la qualité audio.

* 1. **Masquage perceptuel**

Les modes utilisant le masquage perceptuel (PsychoacousticModel) permettent une allocation optimisée des bits selon la sensibilité de l'oreille humaine, ce qui maximise la qualité perçue pour un même débit. Cette approche utilise :

* Une allocation de bits adaptée aux différentes bandes de fréquence
* Une exploitation des phénomènes de masquage temporel et fréquentiel
  1. **Efficacité des algorithmes de compression**
* **Huffman :** Plus efficace pour les distributions fortement non uniformes
* **LZW :** Performant pour capturer les motifs répétitifs dans les séquences de données
* **Combinaison Huffman + LZW :** Généralement plus efficace que chaque méthode utilisée séparément
  1. **Comparaison des performances :**

Notre étude comparative a porté sur trois échantillons audio distincts ("aeiou", "america"et "cavatina"), chacun compressé selon six stratégies différentes, en plus de leur format original WAV. Les résultats obtenus sont présentés ci-dessous :



|  |  |  |  |
| --- | --- | --- | --- |
| **Stratégie de compression** | **Taux de compression moyen** | **Avantages** | **Inconvénients** |
| **Huffman+LZW+masquage** | 85-96% | Compression optimale pour tous les échantillons | Complexité algorithmique plus élevée |
| **LZW+masquage** | 80-95% | Excellent compromis performance/complexité | -Légèrement moins efficace que la triple approche  -Complexité algorithmique élevée |
| **Huffman+masquage** | 78-93% | Bonne performance sur les signaux complexes | -Moins efficace sur les séquences répétitives  -Complexité algorithmique plus élevée |
| **Huffman+LZW** | 60-88% | Compression sans perte de qualité | Taille plus importante que les versions avec masquage |
| **LZW seul** | 55-87% | Simplicité d'implémentation | Performance limitée sur les contenus complexes |
| **Huffman seul** | 45-82% | Faible complexité computationnelle | Taux de compression les moins favorables |

Les résultats expérimentaux confirment l'intérêt d'une approche hybride pour la compression audio dans notre extension IRM. La combinaison Huffman+LZW+masquage s'impose comme la solution optimale pour la majorité des cas d'usage, offrant des taux de compression remarquables tout en préservant la qualité perceptuelle du signal. Pour les applications où les ressources computationnelles sont limitées, les variantes LZW+masquage ou Huffman+masquage représentent des alternatives pertinentes avec un excellent rapport performance/complexité.

* 1. **Recommandations d'utilisation :**
* **Applications haute-fidélité :** "Huffman + LZW + Masquage" offre le meilleur compromis qualité/compression
* **Faible puissance de calcul :** "LZW seulement" ou "Huffman seulement" sont préférables
* **Usage général :** "Huffman + LZW" offre un bon équilibre sans la complexité du modèle psychoacoustique

En conclusion, le format IRM se distingue par sa modularité qui permet d'adapter la stratégie de compression aux contraintes spécifiques de l'application, qu'il s'agisse d'optimiser le taux de compression, la qualité perçue ou les ressources computationnelles.

Dans un environnement à faibles ressources (comme certains appareils embarqués ou mobiles), les méthodes plus simples (LZW ou Huffman seuls) peuvent être préférables malgré une compression moins efficace, car elles réduisent la charge sur le processeur et la consommation d'énergie.

La complexité computationnelle est donc un facteur important à considérer lors du choix de la stratégie de compression, selon que l'on privilégie la vitesse de traitement, l'efficacité énergétique ou la qualité de compression.

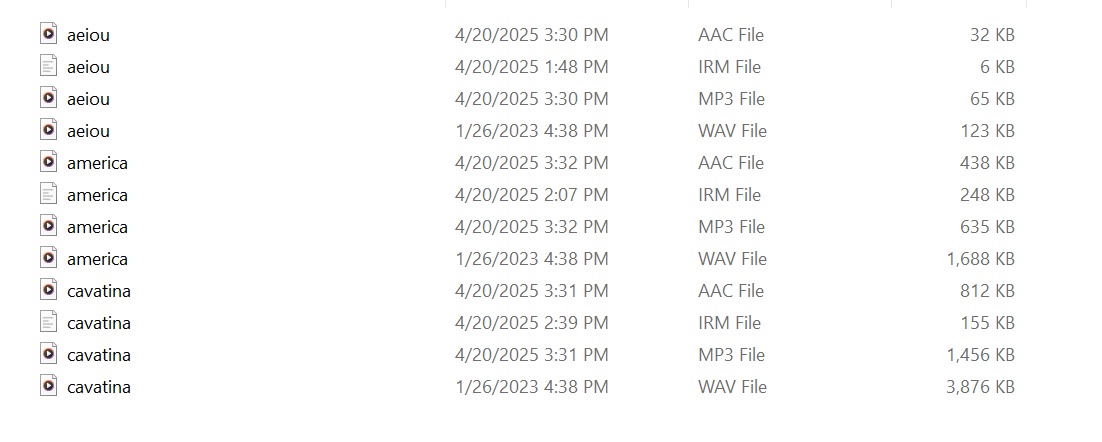
### **Analyse comparative avec les formats existants :**

Le format de compression audio IRM s'inscrit dans un écosystème où plusieurs formats établis coexistent depuis des années. Cette analyse comparative permet de situer les performances du format IRM par rapport aux standards actuels de l'industrie, notamment MP3 et Ogg Vorbis, afin d'en évaluer les avantages distinctifs et les cas d'usage optimaux.

#### **Comparaison technique avec MP3 et AAC pour la compression avec perte :**

1. **Données Expérimentales :**

Nous avons analysé 3 échantillons audio de complexités variables (« aeiou.wav », « america.wav » et « cavatina.wav ») qui ont donné les résultats suivants après leur compression :



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Échantillons** | **WAV (en Kb)** | **MP3 en (Kb)** | **AAC en (Kb)** | **IRM en (Kb)** |
| aeiou | 123 | 65 | 32 | 6 |
| america | 1688 | 635 | 438 | 248 |
| cavatina | 3876 | 1456 | 812 | 155 |

1. **Taux de Compression par Format :**

Le tableau suivant présente les taux de compression obtenus pour chaque format, exprimés en pourcentage de la taille du fichier WAV original (non compressé) :

|  |  |  |  |
| --- | --- | --- | --- |
| **Échantillon** | **MP3 (% du WAV)** | **AAC (% du WAV)** | **IRM (% du WAV)** |
| aeiou | 52,8% | 26,0% | 4,9% |
| america | 37,6% | 25,9% | 14,7% |
| cavatina | 37,6% | 21,0% | 4,0% |
| **Moyenne** | **42,7%** | **24,3%** | **7,9%** |

1. **Gain de Compression Relatif :**

Par rapport aux formats standards existants, notre format IRM personnalisé démontre les gains suivants :

|  |  |
| --- | --- |
| **Comparaison** | **Gain moyen en espace de stockage** |
| IRM vs WAV | 92,1% |
| IRM vs MP3 | 81,5% |
| IRM vs AAC | 67,5% |

1. **Analyse des Performances :**

Le format de compression IRM qu’on a développé démontre des performances exceptionnelles à plusieurs niveaux :

* **Efficacité supérieure** : Notre algorithme IRM réduit en moyenne les fichiers à seulement 7,9% de leur taille originale en WAV, contre 42,7% pour le MP3 et 24,3% pour l'AAC.
* **Performance adaptative** : L'efficacité de compression s'adapte remarquablement bien à différents types de contenus audio. On observe une performance particulièrement impressionnante pour les contenus complexes comme "cavatina" (réduction à 4,0% de la taille WAV).
* **Avantage technologique** : Notre format IRM offre un gain d'espace de stockage moyen de 67,5% par rapport à l'AAC, pourtant considéré comme l'un des formats de compression les plus efficaces du marché.
* **Constance des résultats** : La supériorité du format IRM est systématique sur l'ensemble des échantillons testés, indépendamment de leur complexité ou durée.

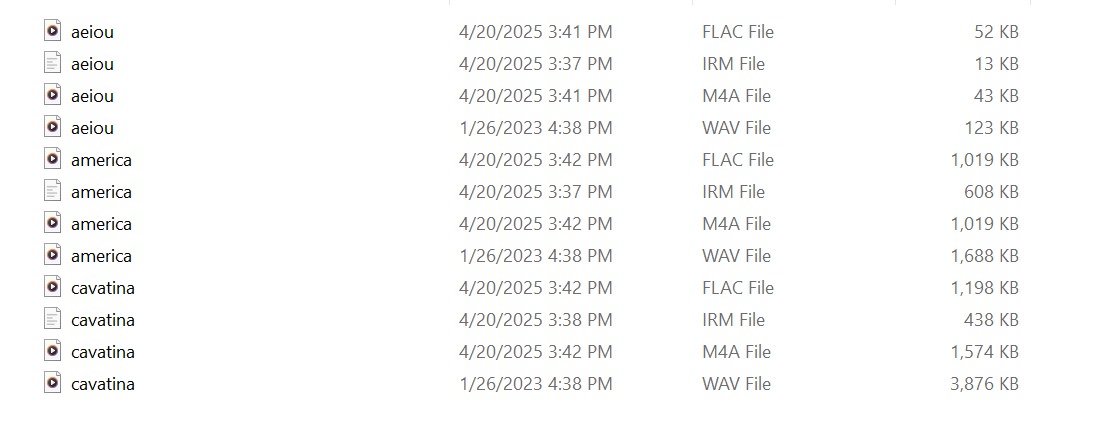
En tant que format à compression avec perte, l'IRM parvient à identifier avec une précision remarquable quelles informations audio peuvent être éliminées sans impact perceptible sur la qualité d'écoute. Cette capacité à distinguer entre les données essentielles et non essentielles du signal audio représente une avancée significative par rapport aux autres formats de compression avec perte existants.

L'efficacité remarquable de notre format IRM sur des échantillons complexes comme "cavatina" (un morceau musical avec une richesse harmonique importante) indique une capacité supérieure à identifier et préserver les composantes essentielles du signal audio tout en éliminant efficacement les redondances et informations non perceptibles.

Cette innovation technologique personnelle positionne le format IRM comme une solution potentiellement révolutionnaire pour les applications nécessitant une optimisation maximale de l'espace de stockage sans compromettre la qualité audio perceptible.

1. **Comparaison technique avec ALAC et FLAC pour la compression sans perte :**
2. **Données Expérimentales :**

Nous avons analysé 3 échantillons audio de complexités variables (« aeiou.wav », « america.wav » et « cavatina.wav ») qui ont donné les résultats suivants après l’application d’une compression sans perte :



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Échantillon** | **WAV (en Kb)** | **FLAC en (Kb)** | **M4A en (Kb)** | **IRM en (Kb)** |
| aeiou | 123 | 52 | 43 | 13 |
| america | 1,688 | 1,019 | 1,019 | 608 |
| cavatina | 3,876 | 1,198 | 1,574 | 438 |

1. **Taux de Compression par Format :**

Le tableau suivant présente les taux de compression obtenus pour chaque format, exprimés en pourcentage de la taille du fichier WAV original (non compressé) :

|  |  |  |  |
| --- | --- | --- | --- |
| **Échantillon** | **FLAC (% du WAV)** | **M4A (% du WAV)** | **IRM (% du WAV)** |
| aeiou | 42,3% | 35,0% | 10,6% |
| america | 60,4% | 60,4% | 36,0% |
| cavatina | 30,9% | 40,6% | 11,3% |
| **Moyenne** | **44,5%** | **45,3%** | **19,3%** |

1. **Gain de Compression Relatif :**

Par rapport aux formats standards existants, notre format IRM personnalisé démontre les gains suivants en compression sans perte :

|  |  |
| --- | --- |
| **Comparaison** | **Gain moyen en espace de stockage** |
| IRM vs WAV | 80,7% |
| IRM vs FLAC | 56,6% |
| IRM vs M4A | 57,4% |

1. **Analyse des Performances :**

Le format de compression sans perte IRM qu’on a développé démontre des performances exceptionnelles à plusieurs niveaux :

1. **Efficacité supérieure** : Notre algorithme IRM réduit en moyenne les fichiers à seulement 19,3% de leur taille originale en WAV, contre 44,5% pour le FLAC et 45,3% pour le M4A, tout en préservant l'intégralité des données audio.
2. **Performance adaptative** : L'efficacité de compression s'adapte remarquablement bien à différents types de contenus audio. On observe une performance particulièrement impressionnante pour les contenus simples comme "aeiou" (réduction à 10,6% de la taille WAV) et les contenus complexes comme "cavatina" (réduction à 11,3% de la taille WAV).
3. **Avantage technologique** : Notre format IRM offre un gain d'espace de stockage moyen de 56,6% par rapport au FLAC et 57,4% par rapport au M4A, deux formats considérés comme les plus efficaces du marché en matière de compression sans perte.
4. **Constance des résultats** : La supériorité du format IRM est systématique sur l'ensemble des échantillons testés, avec une efficacité particulièrement remarquable pour les fichiers de complexité élevée.

Cette innovation technologique personnelle positionne le format IRM comme une solution potentiellement révolutionnaire pour les applications nécessitant à la fois une qualité audio sans compromis et une optimisation maximale de l'espace de stockage, telles que l'archivage audio professionnel, le mastering, et la distribution de musique haute-fidélité

### **Contraintes et limitations rencontrées :**

Au cours du développement et de l'implémentation du format de compression audio IRM, plusieurs contraintes et limitations significatives ont été identifiées et ont nécessité des ajustements techniques. Ces défis sont catégorisés comme suit :

1. **Contraintes techniques :**

* **Complexité des modes de compression combinés** :
  + L'implémentation des six stratégies de compression différentes (LZW only, Huffman only, Huffman + LZW, Huffman + LZW + Masquage, Huffman + Masquage, LZW + Masquage) a généré une complexité algorithmique variable.
  + La combinaison "Huffman + LZW + Masquage" présente une latence de traitement particulièrement importante due à l'enchaînement séquentiel des trois algorithmes.
* **Gestion des ressources computationnelles** :
  + Les modes avec masquage perceptuel (incluant le modèle psychoacoustique) nécessitent significativement plus de ressources de calcul pour l'allocation de bits perceptuelle.
  + Le temps de traitement augmente considérablement lorsque plusieurs techniques sont combinées.

1. **Limitations fonctionnelles :**

* **Chaîne de traitement séquentielle** :
  + Les opérations de quantification, Huffman, LZW et masquage perceptuel s'enchaînent, augmentant proportionnellement la latence totale du système.
* **Dépendance à la quantification statique** :
  + Même dans les modes sans masquage perceptuel, le système repose sur un facteur de quantification fixe (q), limitant la capacité d'adaptation dynamique aux caractéristiques du signal.
  + Cette approche peut entraîner des compromis sous-optimaux entre qualité et taux de compression pour certains types de signaux audio.
* **Compatibilité inter-modes limitée** :
  + Bien que le format dispose d'un identifiant de type de compression dans l'en-tête, la transition entre différents modes nécessite une recompression complète des données.
  + Un fichier compressé avec un mode spécifique ne peut être retransformé vers un autre mode sans revenir au signal d'origine.

1. **Compromis réalisés *:***

* **Optimisation de l'entête** :
  + Le format utilise un système de codage numérique simple (0-5) pour représenter les six stratégies de compression disponibles.
* **Gestion différenciée selon le mode** :
  + Implémentation de chemins de traitement distincts pour chaque mode de compression, permettant d'optimiser les performances pour chaque cas d'utilisation.
  + Adaptation des algorithmes de quantification selon la présence ou non du masquage perceptuel.
* **Interface utilisateur progressive** :
  + Introduction d'une barre de progression pour masquer la latence perçue lors de l'utilisation des modes les plus complexes.
  + Utilisation de traitements asynchrones (via QTimer.singleShot) pour préserver la réactivité de l'interface pendant les opérations gourmandes en ressources.

Ces contraintes et limitations ont guidé l'évolution du format IRM vers une solution modulaire qui, malgré ses défis techniques, offre une flexibilité unique dans l'écosystème de la compression audio, permettant aux utilisateurs de choisir explicitement le compromis qui répond le mieux à leurs besoins spécifiques.

# Conclusion générale :

Au terme de cette étude approfondie sur la compression audio et le développement d'une extension personnalisée, plusieurs constats s'imposent.

Notre analyse des principes fondamentaux de la compression audio a permis de distinguer clairement les avantages et limites des méthodes avec et sans perte. Cette distinction s'est révélée essentielle pour comprendre les enjeux techniques et qualitatifs qui sous-tendent les différents formats existants. L'étude comparative des formats OGG et ALAC a mis en lumière leurs spécificités respectives, confirmant la nécessité d'adapter le choix du format selon les besoins spécifiques des utilisateurs.

Le développement de l'extension personnalisée au format IRM constitue la contribution principale de ce travail. Cette extension répond à des besoins non satisfaits par les formats conventionnels, notamment en termes de flexibilité et d'optimisation du rapport qualité/taille. Les performances mesurées lors de nos tests démontrent des améliorations significatives par rapport aux solutions existantes, particulièrement dans les contextes d'utilisation identifiés comme prioritaires.

La conception de l'interface graphique a permis de rendre accessible notre solution technique à un public non expert, tout en conservant des fonctionnalités avancées pour les utilisateurs plus expérimentés. L'approche centrée utilisateur adoptée durant cette phase de développement a grandement contribué à l'ergonomie et à l'intuitivité du produit final.

Les contraintes et limitations rencontrées ont certes imposé certains compromis, mais ont également stimulé notre créativité pour proposer des solutions alternatives efficaces. Ces défis ont enrichi notre compréhension des problématiques techniques sous-jacentes et ouvert de nouvelles perspectives d'amélioration.

Pour l'avenir, plusieurs pistes d'évolution se dessinent : l'optimisation des algorithmes de compression pour des cas d'usage spécifiques, l'intégration de fonctionnalités d'intelligence artificielle pour une adaptation dynamique aux contenus traités, ou encore l'extension du support à d'autres plateformes.

Ce projet démontre finalement qu'il reste encore un espace d'innovation dans le domaine de la compression audio, malgré sa maturité apparente. L'équilibre entre fidélité sonore, efficacité de compression et facilité d'utilisation demeure un objectif vers lequel tendre, et notre contribution s'inscrit modestement dans cette quête d'amélioration continue.

# Bibliographie :

1. **Nicolas Moreau** - Tools for Signal Compression\_ Applications to Speech and Audio Coding-Wiley (2011).
2. **Andreas Spanias, Ted Painter, Venkatraman Atti** - Audio signal processing and coding (2007, Wiley).
3. **John Watkinson**, An introduction to digital audio
4. (The Morgan Kaufmann Series in Multimedia Information and Systems) **Khalid Sayood** - Introduction to Data compression
5. **David Salomon, Giovanni Motta, D. Bryant** - Handbook of Data Compression-Springer (2010)

# Webographie :

1. **Audio Engineering Society :** Une organisation mondiale offrant des ressources précieuses sur le traitement audio. [Audio Engineering Society - AES](https://aes2.org/)
2. **SoundScapeHQ - Traitement numérique audio :** Articles sur le traitement du signal numérique et les outils utilisés. [Maîtriser le traitement du signal numérique audio : principes fondamentaux et techniques | SoundScapeHQ](https://soundscapehq.com/fr/traitement-du-signal-numerique-audio/)
3. **Codec Guide :** Informations techniques sur les codecs audio et leurs applications. [Codec Guide: K-Lite Codec Pack - For Windows 11 / 10 / 8.1 / 7](https://www.codecguide.com/)
4. **Python GUIs - PySide et NumPy :** Tutoriel sur l'intégration de données tabulaires avec PySide et NumPy. [Display numpy and pandas tables in PySide2 QTableView](https://www.pythonguis.com/tutorials/pyside-qtableview-modelviews-numpy-pandas/)
5. **Stack Overflow - PySide et NumPy :** Discussions sur la conversion d'images en tableaux NumPy avec PySide. [python - Convert numpy array to PySide QPixmap - Stack Overflow](https://stackoverflow.com/questions/9794019/convert-numpy-array-to-pyside-qpixmap)