Ce projet a pour objectif de concevoir un logiciel multimédia multiplateforme, inspiré de solutions existantes comme *Kodi*.
L'application permettra de lire, organiser et diffuser des contenus audio et vidéo tout en offrant une interface moderne et personnalisable.
Elle intégrera également un système d'extensions afin de permettre aux utilisateurs d'ajouter de nouvelles fonctionnalités ou sources de contenu.

L'enjeu principal est de proposer un outil à la fois performant, fiable et extensible, capable de fonctionner sur différents systèmes d'exploitation (Windows, Linux, macOS).
Dans cette optique, plusieurs langages de programmation ont été étudiés afin d'identifier celui offrant le meilleur équilibre entre performance, sécurité, maintenabilité et portabilité.

Langage	Points forts	Points faibles	Verdict
C++	- Performances excellentes - Écosystème multimédia très mature (FFmpeg, Qt, etc.) - Très flexible et bas niveau	- Gestion mémoire manuelle (risque de crashs) - Compilation complexe et lente - Maintenance difficile	Solide techniquement, mais vieillissant et coûteux à maintenir
Python	- Facile à utiliser et à étendre - Parfait pour le scripting et les plugins - Écosystème riche	- Lent pour la lecture multimédia - Mauvaise gestion du multithreading (GIL) - Non adapté pour le cœur du moteur	Excellent pour les extensions, pas pour le moteur principal
JavaScript / TypeScript	- Développement UI rapide (HTML/CSS) - Multiplateforme (Electron, Tauri) - Large communauté	- Performances faibles - Consommation mémoire élevée - Moins adapté pour le traitement bas niveau	Bon choix pour l'interface, pas pour le moteur multimédia
C# / .NET	- Bon équilibre perf/productivité - Bon support multiplateforme (.NET MAUI) - Syntaxe claire	- Runtime .NET obligatoire - Écosystème multimédia limité - Moins de contrôle bas niveau	Correct, mais pas optimal pour un moteur léger et extensible

Rust 🦀 *(choisi)*	- Performances proches du C++ - Sécurité mémoire garantie - Multithreading sûr - Écosystème moderne (gstreamer-rs, Tauri, Slint) - Excellent interfaçage (C, Python, etc.)	- Apprentissage plus complexe - Compilation parfois lente - Écosystème UI encore jeune	Meilleur compromis : rapide, sûr, moderne et multiplateforme

En conclusion, après comparaison des différentes options, Rust s'impose comme l'un des choix les plus pertinents pour ce projet de lecteur multimédia multiplateforme. Il combine les performances et le contrôle d'un langage natif comme C++, tout en garantissant une sécurité mémoire et une fiabilité que peu d'autres langages offrent.
Son écosystème en pleine croissance, notamment autour de GStreamer, Tauri ou Slint, permet aujourd'hui de concevoir des applications robustes, modernes et maintenables sur le long terme.

Le choix de Rust reflète donc une volonté d'allier performance, sécurité et modernité, tout en assurant une base technique durable pour faire évoluer le projet dans le temps.