Dossier de modélisation

LARRYTHMIQUE

29 novembre 2024

Laurian JAMIN & Corentin RICHARD

Avant-propos

Le rôle de ce document est d'être un support pour la modélisation de notre application de reconnaissance des audios « Larrythmique », il doit nous permettre de réfléchir, créer et utiliser cette modélisation afin de poursuivre le développement. Il intègre également un choix d'outils pour la modélisation ainsi qu'une étude de la législation concernant les droits d'utilisation de musiques dans des applications publiques.

Table of Contents

I. Outils de modélisation logiciel

Dans un premier temps, il est crucial de sélectionner un outil adapté à la modélisation de notre application de reconnaissance audio. Pour ce faire, nous devons d'abord identifier et définir les critères prioritaires, en tenant compte des spécificités et des contraintes du contexte dans lequel nous opérons. Une fois ces critères clairement établis, nous pourrons analyser et comparer les différents outils disponibles sur le marché, afin de choisir celui qui répond le mieux à nos exigences et de prendre une décision éclairée.

A. Critères importants

Pour situer la SAE dans son contexte, nous travaillons actuellement sur la modélisation au sein d'une équipe composée de deux personnes, avec un temps limité pour mener à bien ce projet.

Tout d'abord, nos parcours académiques étant différents, nous ne disposons pas des mêmes compétences ni de la même expérience avec les outils disponibles. Il est donc essentiel de sélectionner un outil qui soit à la fois intuitif et rapide à prendre en main. Cela nous permettra d'optimiser notre efficacité tout en assurant que chaque membre de l'équipe puisse contribuer pleinement au projet, sans rencontrer de difficultés techniques.

Ensuite, étant donné que nous travaillons à partir de deux postes distincts, il est crucial de disposer d'un outil collaboratif qui permette un accès facile aux données depuis n'importe lequel de ces postes. Cependant, il est également important que cet outil ne limite pas le nombre d'éléments qu'il est possible de créer, afin de répondre pleinement aux besoins du projet.

Sur la base de ces critères, nous pourrons ensuite évaluer les outils disponibles et choisir celui qui correspond le mieux à nos exigences.

B. Evaluation des outils

Afin de synthétiser le comparatif, nous ne présenterons pas tous les outils que nous avons pu voir dans notre tableau mais nous avons pu voir Lucidchart, Miro, Draw.io, Microsoft Visio, Figma, Creately, Cacoo, PlantUML, Visual Paradigm, StarUML, Gliffy, Enterprise Architect, Balsamiq, yEd Graph Editor, IBM Rational Software Architect.

Outil	Avantages	Inconvénients
Lucidchart	- **Collaboration** en temps réel - **Accessible** via navigateur - Interface intuitive	- **Limitation** du nombre de diagrammes dans la version gratuite - Nécessite une connexion internet
Miro	- **Collaboration** fluide - **Accessible** sur tous les postes via navigateur - **Simple** et visuel	- Moins adapté aux modèles techniques complexes - **Limitations** dans la version gratuite
Draw.io	- **Gratuit** - **Accessible** en ligne ou localement - Intégration avec **Google Drive** ou OneDrive - **Outils UML**	- Collaboration en temps réel moins fluide que les outils premium
Microsoft Visio	- Puissant pour des modèles complexes - Intégration avec Microsoft 365 - **Compatible** local et cloud	- **Coût élevé** - **Courbe d'apprentissage** plus importante
Figma	- **Collaboration** en temps réel - **Accessible** depuis n'importe quel poste - Interface moderne et **intuitive**	- Principalement **orienté design** graphique, mais utilisable pour modélisation - Fonctionnalités avancées **payantes**
Cacoo	- **Collaboration** en ligne - Interface **intuitive** - Bonne **accessibilité** sur navigateur	- Version gratuite **limitée** - Moins complet pour des projets techniques
Creately	- **Collaboration** efficace - **Accessible** depuis n'importe quel poste - **Interface** simple à prendre en main	- Fonctionnalités **limitées** dans la version gratuite - Peu adapté aux projets très complexes

C. Choix

Nous avons décidé d'utiliser **Draw.io** pour notre projet de modélisation. Ce choix repose sur plusieurs raisons essentielles :

1. **Gratuité et accessibilité** : Draw.io est entièrement gratuit et peut être utilisé à la fois en ligne et hors ligne, ce qui répond à notre besoin de flexibilité pour travailler sur différents postes.
2. **Outils UML intégrés** : L'outil propose des fonctionnalités spécifiques pour créer des diagrammes UML, qui sont particulièrement adaptés à notre projet. Cela nous permet de représenter efficacement les différents aspects de notre application de reconnaissance audio.
3. **Simplicité d'utilisation** : L'interface est intuitive, ce qui facilite la prise en main rapide, un critère essentiel compte tenu de notre temps limité.
4. **Collaboration et intégration** : Bien qu'il ne soit pas aussi performant que certains outils premium pour la collaboration en temps réel, Draw.io permet de partager et de synchroniser facilement les fichiers via des services comme Google Drive ou OneDrive. Cela garantit que nous pouvons récupérer et modifier les données depuis n'importe quel poste.

En somme, Draw.io répond à nos principaux critères en combinant simplicité, fonctionnalités UML, et accessibilité, tout en restant gratuit, ce qui en fait l'outil idéal pour notre contexte.

II. Modélisation

A. Présentation

Notre objectif est de reprendre les principes de base de Shazam pour son ergonomie ainsi que ses fonctionnalités principales tout en gardant l'extension possible pour ajouter de nouvelles fonctionnalités.

Cette modélisation doit donc être le reflet de cet aspect avec en restant compatible avec le plus grand choix de langage et de contexte.

Enfin l'objectif principal de cette modélisation est de servir de plan à la suite du projet.

B. Ordre

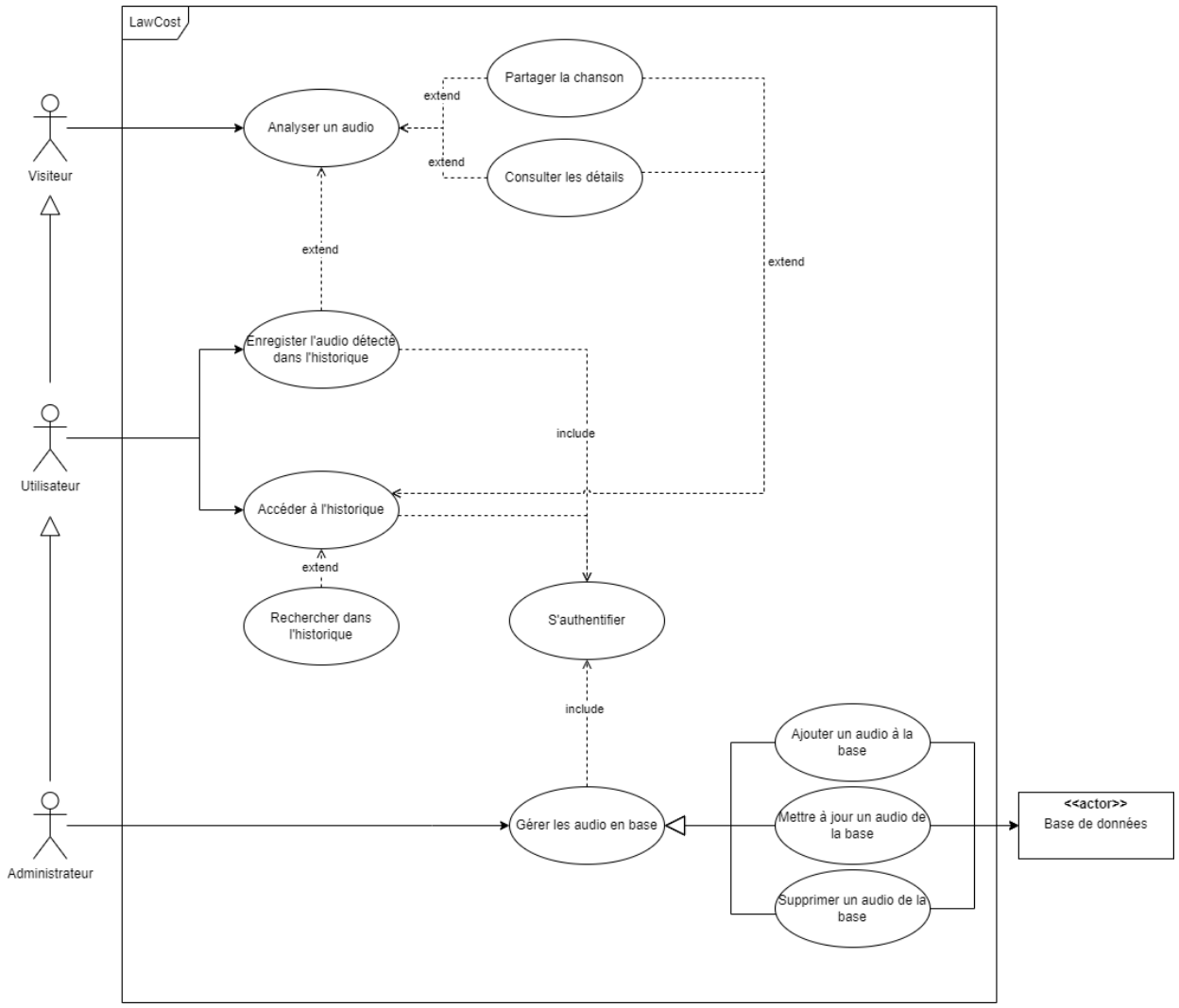
L'ordre suivi pour la modélisation s'appuie sur une logique progressive, partant des aspects fonctionnels globaux pour aller vers des détails techniques et structurels. Le **diagramme de cas d'utilisation** constitue une première étape essentielle, car il permet d'identifier les interactions entre les acteurs et le système, posant ainsi les bases des besoins fonctionnels. Ensuite, le **diagramme d'activité** détaille les processus et les flux logiques associés aux cas d'utilisation, offrant une vue plus opérationnelle.

Le diagramme d'état, focalisé sur un élément clé comme la musique, apporte une compréhension des transitions et des comportements spécifiques, ce qui aide à compléter la modélisation comportementale. Le **diagramme de séquence** vient ensuite pour détailler les interactions entre les objets ou composants dans des scénarios précis, enrichissant les cas d'utilisation avec des échanges concrets.

Le **diagramme d'architecture** s'intègre logiquement à ce stade, car il structure les composants du système et montre leurs relations globales, en s'appuyant sur les interactions précédemment définies. Le **diagramme de classes** en découle naturellement, traduisant les concepts identifiés en structures concrètes pour le développement. Enfin, le **MCD** est une conclusion cohérente, représentant les données nécessaires et s'assurant qu'elles répondent aux besoins fonctionnels et structurels du système.

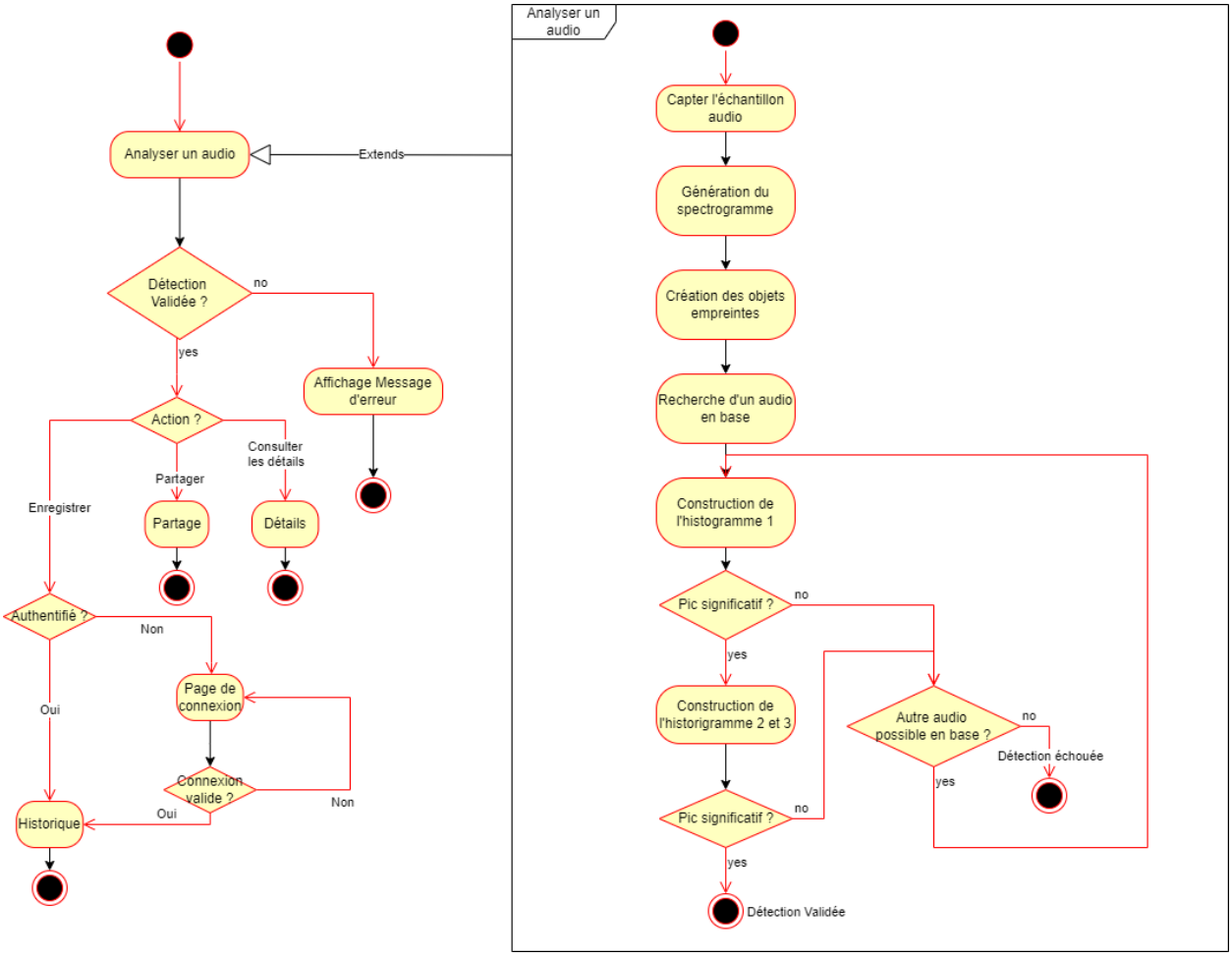
Cet enchaînement justifie une approche claire et organisée, allant des besoins utilisateur vers une modélisation prête à être implémentée.

C. Diagramme de cas d’utilisation



Le diagramme met en évidence trois acteurs principaux, dont l'utilisateur et l'administrateur, ce dernier étant un utilisateur avec des privilèges supplémentaires. L'utilisateur doit obligatoirement s'authentifier pour accéder à des fonctionnalités avancées, comme l'historique des analyses, tandis que l'administrateur peut gérer la base d’audios directement depuis l’application en ajoutant, modifiant ou supprimant des données. La base d’audios joue un rôle central, et sa gestion est essentielle pour garantir la pertinence et la mise à jour du système.

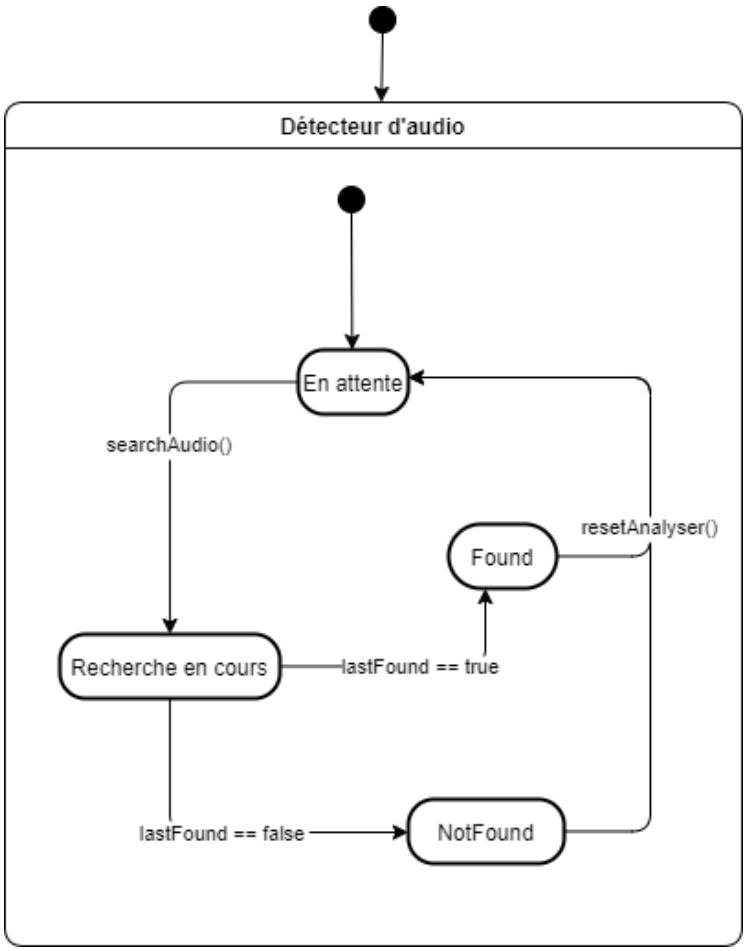
D. Diagramme d’activité du cas « Analyser un audio »



Le diagramme d'activité illustre le processus d’analyse d’un audio, depuis la capture de l’échantillon jusqu’à la validation ou l’échec de la détection. Une fois l’audio analysé, si la détection est valide, l’utilisateur peut consulter les détails, partager l’audio ou l’enregistrer dans l’historique, cette dernière action nécessitant une authentification préalable.

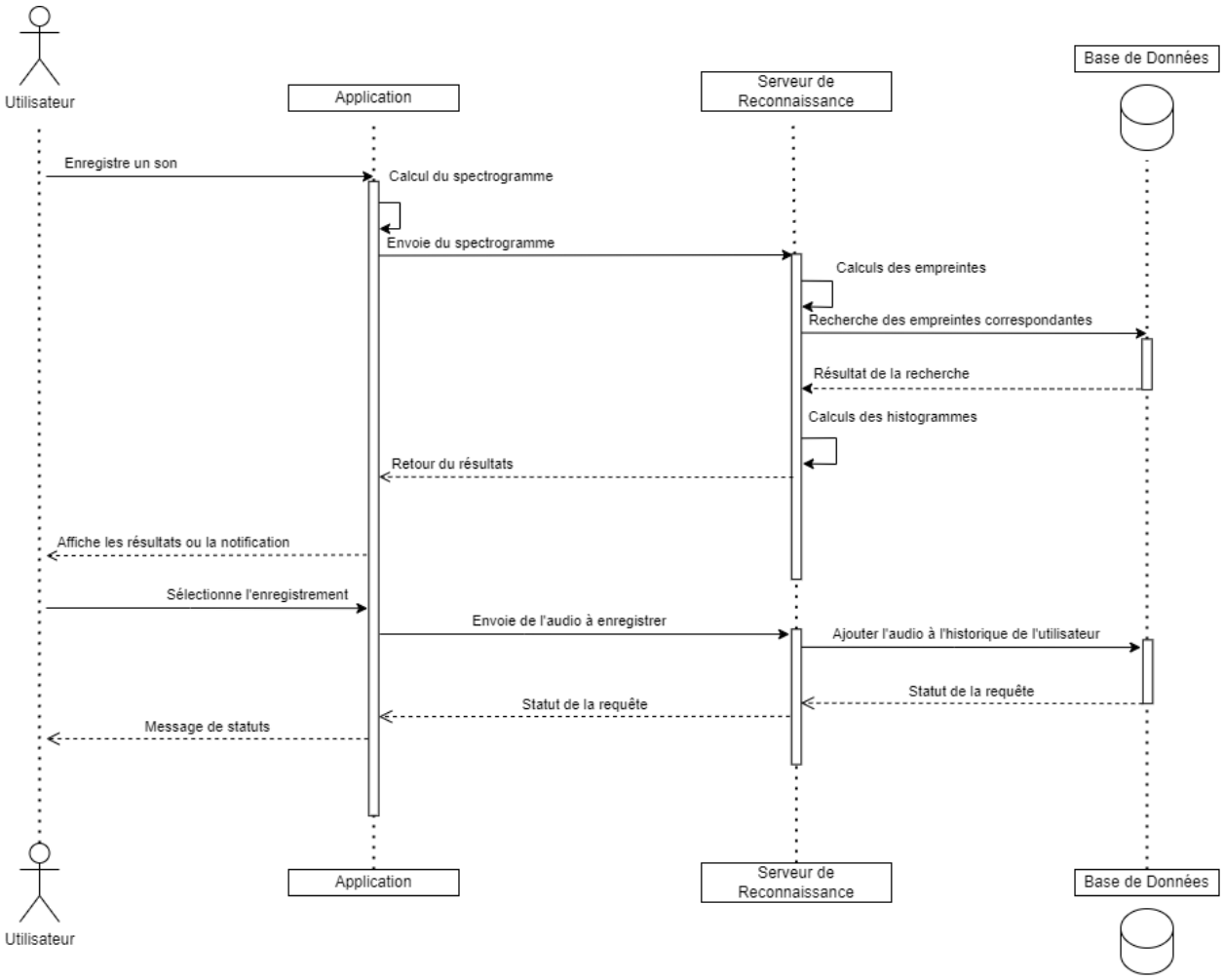
Sur le plan technique, l’analyse repose sur la génération de spectrogrammes et la recherche en base via des histogrammes pour valider la correspondance. Ce processus met en avant l’importance des étapes de vérification pour garantir la fiabilité des résultats. Cependant il n’inclut pas le choix architecturale fait par la suite d’utilisation d’un serveur distant pour effectuer une partie des calculs.

E. Diagramme d’état de l’analyseur



Ce diagramme d’état illustre le fonctionnement du détecteur d’audio. Le système commence dans un état "En attente", prêt à lancer une recherche. Lorsqu’une recherche est initiée via la méthode *searchAudio()*, le détecteur passe à l’état "Recherche en cours" durant laquelle il enregistre l’audio et l’analyse. Une fois la recherche terminée, deux résultats sont possibles : si une correspondance est trouvée *(lastFound == true),* l’état passe à *"Found",* sinon il bascule vers *"NotFound".* Dans les deux cas, un retour à l’état initial "En attente" est possible via la méthode *resetAnalyser().*

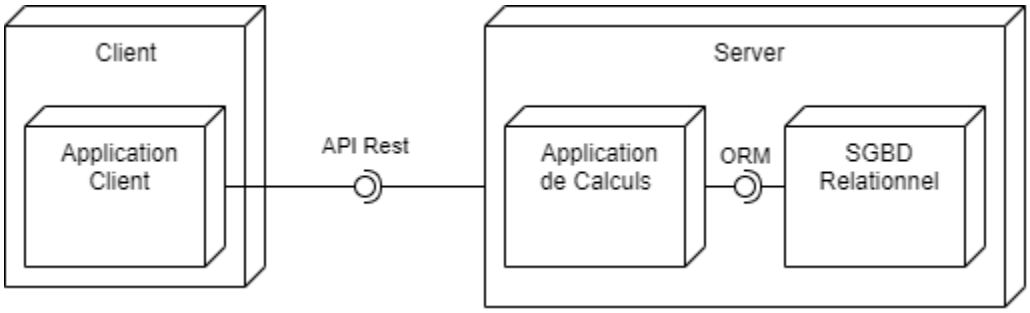
F. Diagramme de Séquence du cas « Analyser un audio »



Dans ce scénario d'analyse d'un audio pour un utilisateur authentifié, le spectrogramme est calculé côté client afin de réduire la charge et le volume des données transmises au serveur via HTTPS, rendant l'envoi plus léger qu'un fichier audio brut. Le serveur de reconnaissance effectue ensuite les calculs tels que la génération d'empreintes et la recherche dans la base de données.

Une fois le résultat obtenu, l'audio peut être enregistré dans l'historique de l'utilisateur via une interaction entre le serveur et la base de données. Ce choix d'architecture représente un compromis : il délègue une partie des calculs au client, ce qui convient aux postes disposant de ressources raisonnables, tout en laissant les opérations critiques au serveur pour assurer la compatibilité avec des plateformes plus légères.

G. Diagramme d’architecture



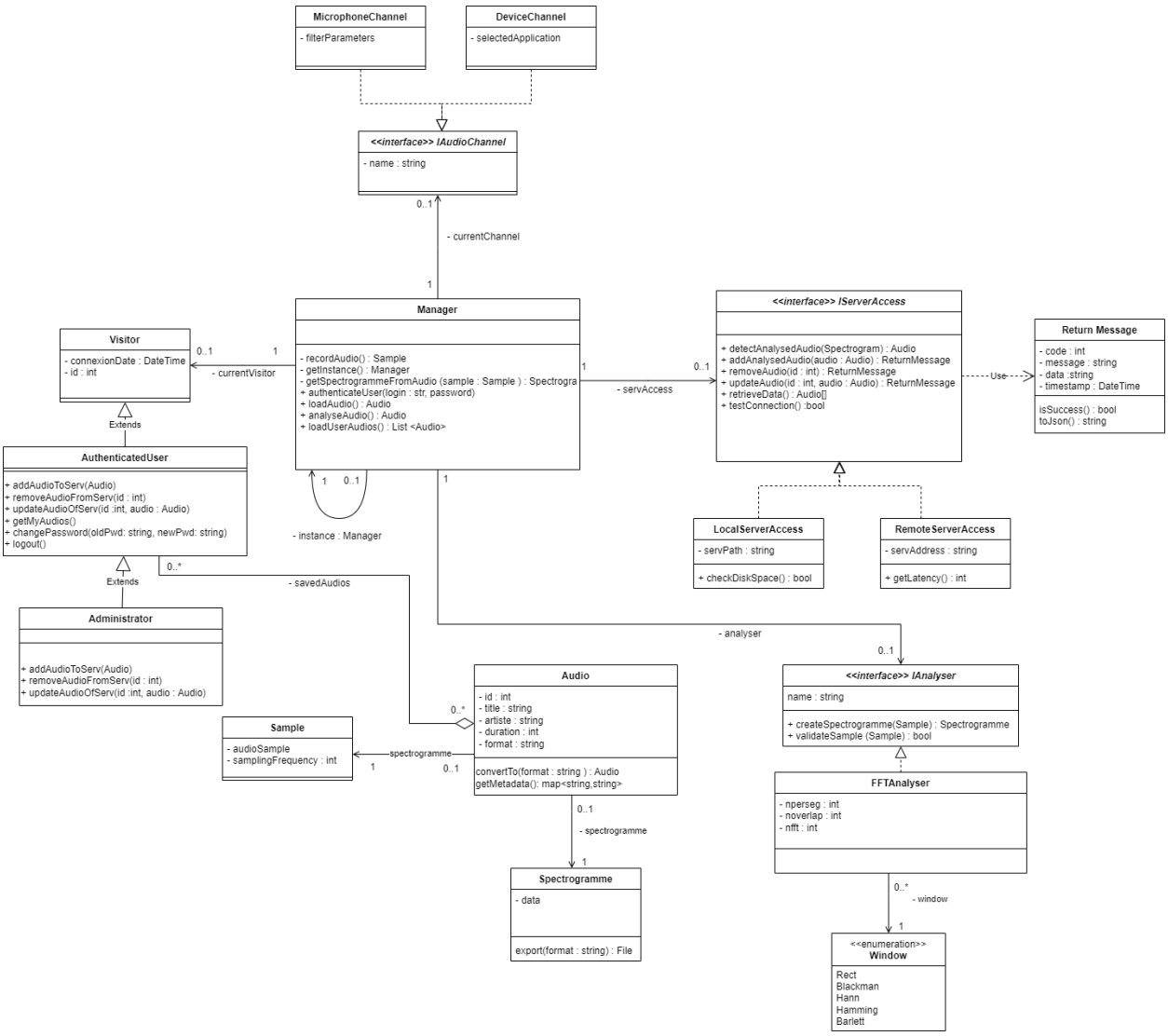
L’architecture proposée repose sur une séparation claire entre le client et le serveur, favorisant une communication sécurisée via des requêtes HTTPS. L’application cliente est responsable de l’interface utilisateur et interagit avec le serveur par le biais d’une API REST.

Côté serveur, une application de calculs est déployée pour gérer les traitements nécessaires et assurer la liaison avec une base de données relationnelle hébergée localement. L’utilisation d’un ORM (Object-Relational Mapping) est privilégiée pour fluidifier les interactions entre l’application et la base de données, simplifiant les requêtes, leur maintenance et éviter les injections de SQL.

L’authentification des utilisateurs repose sur des mots de passe salés puis hachés, renforçant ainsi la sécurité des données sensibles.

Cette architecture requiert donc au moins 2 diagrammes de classes. Un pour le client et un autre côté server. Il faudra également effectuer un MCD pour représenter les données stockées en base.

H. Diagramme de classe client

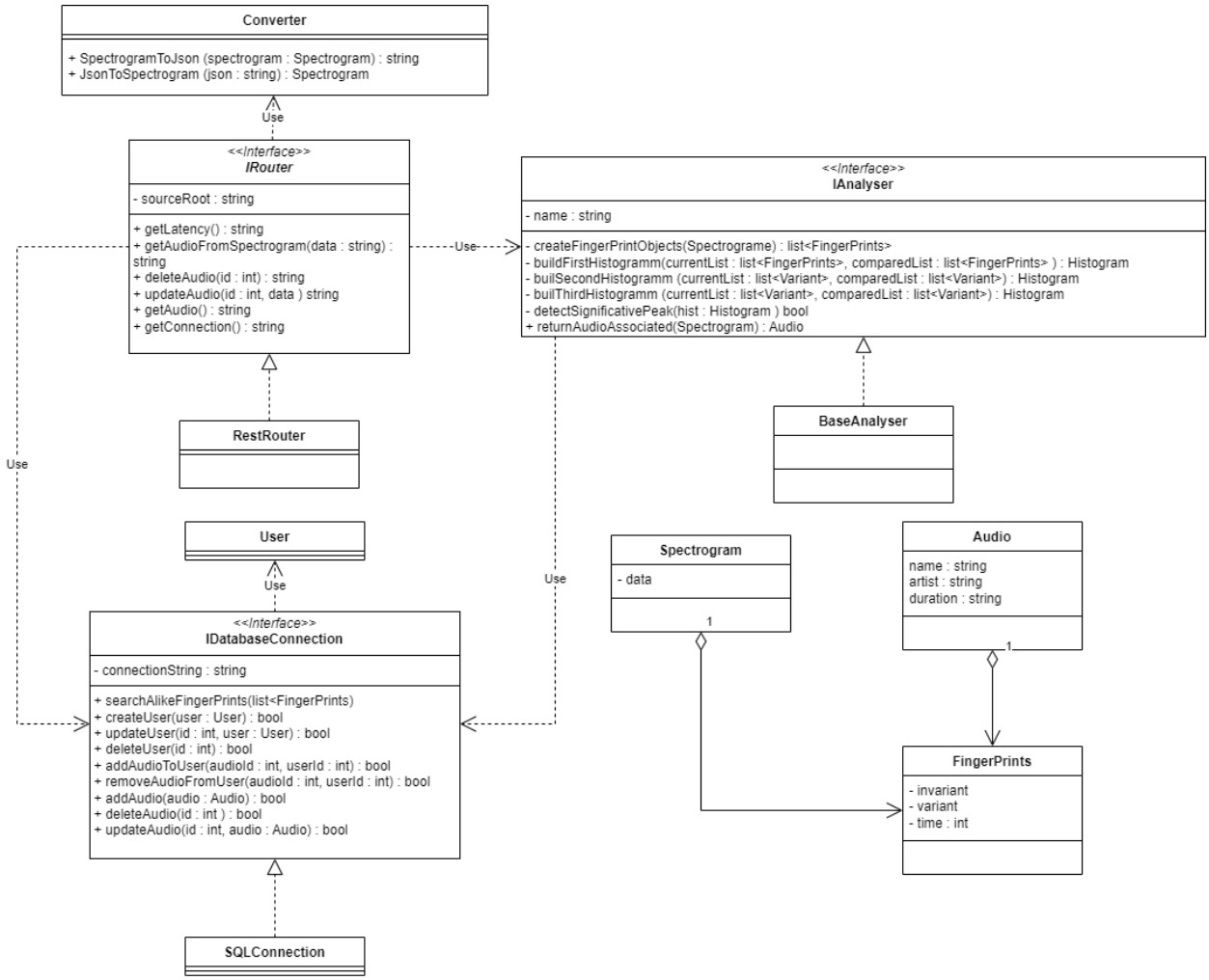


Au cœur du système se trouve la classe Manager, qui orchestre les opérations de gestion des audios, la génération de spectrogrammes et l'authentification des utilisateurs. Les classes Visitor, AuthenticatedUser et Administrator décrivent les différents types d'utilisateurs, avec des permissions croissantes, allant de la consultation à la gestion complète des audios.

Les interfaces, telles que IAudioChannel, IServerAccess, et IAnalyser, assurent la flexibilité et l'extensibilité du système en définissant des comportements communs implémentés par des classes dérivées comme MicrophoneChannel, DeviceChannel, LocalServerAccess, RemoteServerAccess, et FFTAnalyser. La relation entre ces classes met en avant l'importance des protocoles de connexion au serveur et de l'analyse des spectrogrammes pour garantir la validité et la fiabilité des audios traités.

Les classes Audio et Sample sont au cœur des données, avec Spectrogramme jouant un rôle crucial dans la représentation visuelle des échantillons audio. Les méthodes associées à ces classes permettent des opérations telles que la conversion de formats, l'exportation de données, et l'extraction de métadonnées.

I. Diagramme de classe serveur



Le diagramme de classe côté serveur du système met en lumière la structure et les interactions des différents composants essentiels au fonctionnement du serveur. Les principales entités incluent les classes Converter, RestRouter, User, SQLConnection, BaseAnalyser, Spectrogram, Audio, et FingerPrints.

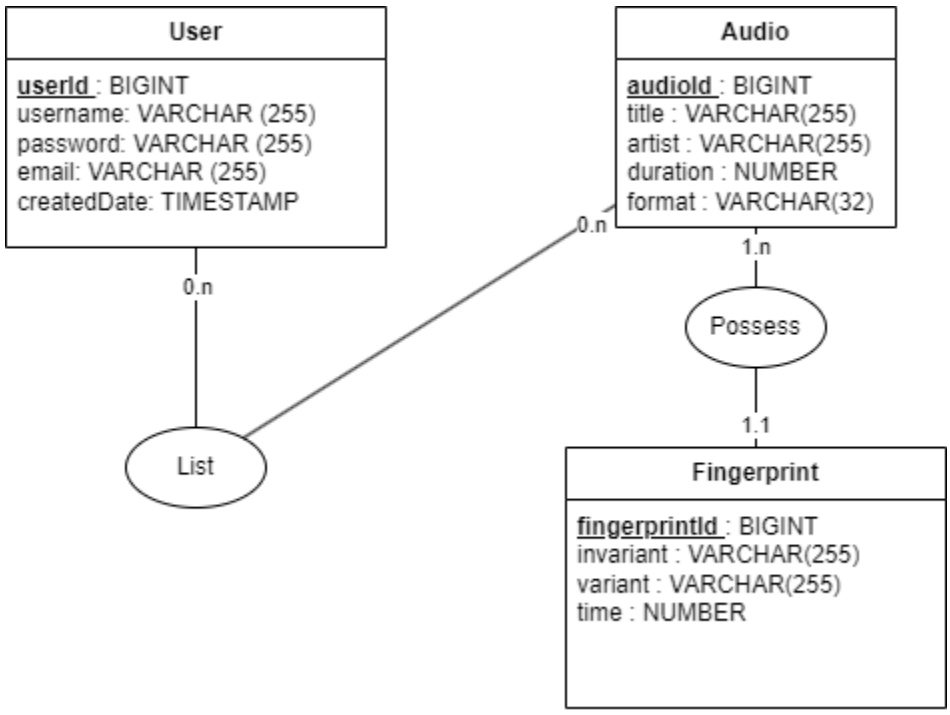
La classe Converter joue un rôle crucial en convertissant les spectrogrammes en JSON et vice versa, facilitant ainsi la manipulation des données au sein du système. RestRouter, implémentant l'interface IRouter, gère le routage des requêtes, permettant d'obtenir des latences, de récupérer des audios à partir de spectrogrammes, et de gérer les connexions.

L'interface IDatabaseConnection, mise en œuvre par SQLConnection, définit les méthodes pour les opérations de base de données, telles que la création, la mise à jour, la suppression des utilisateurs et des audios, ainsi que la recherche de fingerprints similaires. Cette connexion à la base de données est essentielle pour maintenir l'intégrité et la disponibilité des données.

Le BaseAnalyser, en tant qu'implémentation de l'interface IAnalyser, se charge de l'analyse des spectrogrammes, de la création d'objets de fingerprint, et de la construction d'histogrammes pour détecter les pics significatifs, assurant ainsi la fiabilité des détections d'audio.

Les classes Spectrogram, Audio, et FingerPrints représentent respectivement les spectrogrammes, les fichiers audios, et les empreintes digitales de l'audio, avec leurs attributs et méthodes spécifiques permettant de manipuler et d'analyser les données audios.

J. Diagramme de base de données relationnelle (MCD)



Le diagramme de la base de données décrit la relation entre les entités User, Audio, et Fingerprint, où chaque utilisateur peut posséder plusieurs fichiers audios, et chaque audio peut être associé à plusieurs empreintes digitales pour l'analyse. Les attributs principaux incluent l'identifiant unique, le nom d'utilisateur, le mot de passe et le courriel pour les utilisateurs, ainsi que le titre, l'artiste, la durée et le format pour les audios. Les empreintes digitales contiennent des données spécifiques permettant d'identifier et d'analyser les fichiers audios de manière précise.

III. Utilisation d’œuvre musicale dans un programme

Le fait d'utiliser une œuvre musicale dans un programme informatique est régi par le droit d'auteur, qui protège les créations artistiques, y compris les compositions musicales. En France, c'est le Code de la propriété intellectuelle (CPI) qui encadre ces différentes questions.

Dans un premier temps, on distingue les droits patrimoniaux, qui permettent à l'auteur de contrôler l'exploitation de son œuvre, notamment sa reproduction, sa distribution et sa représentation.

Ensuite, nous avons les droits moraux, qui garantissent le respect de l'intégrité de l'œuvre et le droit à la paternité. Ces droits sont perpétuels et imprescriptibles.

Mais alors, avons-nous le droit d'utiliser un morceau de musique dans un programme informatique ? Dans notre cas, pour entraîner un algorithme, il faut normalement l'autorisation des titulaires des droits, sauf si l'œuvre est dans le domaine public (70 ans après la mort de l'auteur) ou si une exception légale s'applique. Sinon, toute reproduction, même temporaire, doit être soumise à l'accord de l'auteur ou de ses ayants droit.

Il existe toutefois des exceptions, bien que celles-ci soient assez limitées. Par exemple, dans le domaine de l'intelligence artificielle, l'utilisation d'œuvres musicales peut être autorisée dans certains pays pour entraîner des IA à des fins scientifiques, mais cela reste soumis à des règles précises et peut exiger une autorisation. En France, grâce à l'article L122-5 du CPI, il est permis d'utiliser des extraits d'œuvres dans un cadre strictement pédagogique, éducatif, clairement délimité et non commercial.

Il y a également la possibilité de recourir à des banques d'œuvres libres, comme Free Music Archive ou Jamendo, pour accéder à des œuvres sous licences ouvertes ou dans le domaine public. Toutefois, il convient de vérifier les législations nationales, car elles peuvent varier.

À l'international, certaines législations abordent aussi ce sujet. Par exemple, la Convention de Berne, un traité qui garantit la protection automatique des œuvres dans 179 pays, ou encore l'accord sur les ADPIC (OMC), un accord sur les aspects des droits de propriété intellectuelle qui touchent au commerce et renforcent les normes internationales en matière de protection.

Références des liens cités

1. **Droit d’auteur et image numérique** : Ministère de l’Économie
2. **La diffusion d’une œuvre protégée** : JurisExpert
3. **Article L122-5 du CPI** : Légifrance
4. **Protection internationale des œuvres** : OMPI
5. **Accord sur les ADPIC** : UNESCO