Rapport final du projet SpotiDut : Serveur de streaming musical sur Raspberry Pi

Maîtrise d'oeuvre
Vanessa BAR - Baptiste BLANCHARD
Pierre DECHENEST - Thibault GIRARD
Axel HEINE

Assistant à maîtrise d'oeuvre
Axel HEINE

Maîtrise d'ouvrage
Stéphane LOHIER

REMERCIEMENTS

Avant d'entamer ce rapport, nous souhaitions remercier toutes les personnes s'étant rendues disponibles et ayant permis la réalisation de ce projet.

Nous remercions donc M. LOHIER pour sa disponibilité, son écoute et ses conseils. Nous le remercions également pour nous avoir fourni le matériel nécessaire à la réalisation du projet.

Nous remercions le corps enseignant qui a nous a permis d'acquérir les compétences nécessaires à la réalisation de ce projet, aussi bien techniques que pratiques, en passant par l'apprentissage des langages de programmation et la gestion de projet.

Nous remercions les enseignants/chercheurs dont M. FORAX et M. PILLIET nous ayant fortement aidé pour la partie Android du projet. Nous remercions également M. CHILOWICZ et M^me^ BEAL pour nous avoir orienté vers ces derniers.

Nous remercions l'IUT qui nous a permis d'obtenir une salle de travail durant toute l'année.

Nous remercions finalement toute autre personne extérieure ayant été force de proposition et de conseils.

SOMMAIRE

INTRODUCTION

Dans le cadre de la deuxième année de DUT Informatique, nous devions réaliser un projet tuteuré par groupe de 3 à 5 personnes. Pour satisfaire notre client, M. LOHIER, nous avons dus mettre en application toutes les connaissances et compétences que nous avons acquises durant notre formation et faire face aux difficultés auxquelles nous avons été confronté.

Ce dossier est un bilan de ce qui a été produit cette année. Nous parlerons plus en détails des différentes réalisations produites et de notre organisation. Ce projet tuteuré a été pour nous l'opportunité de nous investir réellement dans un travail enrichissant et intéressant pour notre future vie professionnelle.

Ce rapport contient l'ensemble des éléments du projet. D'un point de vue technique tout d'abord, nous vous présenterons le cahier des charges tel que nous l'avons imaginé avec M. LOHIER. Nous décrirons le fonctionnement de notre projet dans son ensemble, son architecture et nos choix de conception. Pour terminer la partie technique, nous parlerons des difficultés techniques rencontrées et des solutions apportées.

La deuxième partie de ce rapport a pour objectif de présenter la manière dont nous avons géré le projet. Nous vous expliquerons premièrement comment la communication au sein du projet a été menée. Ensuite, nous verrons comment le projet a été découpé afin de mieux se répartir le travail et comment les missions ont été attribuées. Pour appuyer nos dires, nous joindrons le diagramme de Gantt tel qu'il a été établi. Finalement, nous ferons le point sur les difficultés rencontrées.
Nous espérons que vous prendrez autant de plaisir à lire ce rapport que nous en avons pris durant toute la réalisation de ce projet.

PARTIE I : PRESENTATION DE L'EQUIPE ET DU PROJET

I. L'équipe

Notre équipe, composée de Vanessa BAR, Baptiste BLANCHARD, Pierre DECHENEST, Thibault GIRARD et Axel HEINE, a choisi de réaliser le projet "SpotiDut", encadré par M. LOHIER.

II. Contexte

Avec la dématérialisation et l'essor des nouvelles technologies, on constate la disparition des supports audio. En effet, la musique est de plus en plus fréquemment stockée sur des mémoires flashs et/ou des disques durs, sous forme de simples fichiers MP3, par exemple. Cependant, cette technique est très lourde et cause une perte d'espace de stockage importante : si on possède plusieurs outils, il faut alors copier les fichiers sur chaque plateforme…

De nos jours, les outils ont également pour vocation de regrouper plusieurs outils : avec le téléphone, par exemple, on peut téléphoner, aller sur Internet, écouter de la musique… Ainsi, il paraît nécessaire de développer chaque application sur toutes les plateformes disponibles. En ce qui nous concerne, on constate qu'il est possible de retrouver un lecteur de musiques aussi bien sur un téléphone ou une montre par exemple.

Pour répondre à cette problématique de stockage et cette nécessité de multi- supports, des sites web voient le jour. On peut alors trouver Deezer, YouTube, ou bien Spotify. Ces différents sites permettent tous une écoute illimitée de musiques en ligne, donc, sans stockage : ce sont des sites de streaming.

Cependant, les musiques disponibles sur ce site sont stockées sur des serveurs souvent très onéreux. SpotiDut a pour ambition de récréer un serveur de streaming sur Raspberry Pi, une plateforme simple et peu onéreuse. Il se diffère également des applications déjà existantes puisque chacun pourra posséder son propre serveur chez soi. Il apporte donc une autonomie totale pour son utilisateur.

III. Besoins

L'objectif principal était de créer un serveur local d'écoute de musiques (streaming) ainsi qu'un client permettant leur lecture. Le serveur est hébergé sur une Raspberry Pi à laquelle est connecté un disque dur contenant des musiques. Il permet le stockage des données (musicales et de connexion) et leur envoi.

Le serveur est également connecté à l'application client via un réseau local ou distant. L'application a été créée dans le but de rendre accessible la lecture de musique à l'utilisateur. Elle est constituée d'un player audio et d'un système de recherche. Une des finalités de ce projet était de réduire les coûts en stockage. Il était alors primordial que l'application soit multi-plateformes : elle est alors disponible sur Linux, Windows, Mac et sur Android (optionnel).

Un site web est également mis en ligne afin que les utilisateurs puissent découvrir et comprendre le fonctionnement de cette application. Il permet la création de comptes clients (avec identifiant et mot de passe) rendant disponible le téléchargement gratuit du logiciel. Le site web et le serveur partagent la même base de données.

Au fur et à mesure du projet, de nouvelles fonctionnalités ont été développé avec M. LOHIER. Ainsi, une application d'administration a vu le jour. Elle permet d'ajouter directement des musiques, des albums, des artistes et des utilisateurs au serveur, sans nécessité de posséder des compétences informatiques. Il est également possible d'importer automatiquement en base de données toutes les musiques contenues sur un disque dur connecté à la Raspberry Pi.

Finalement, il était important pour nous de proposer un serveur autonome que chacun puisse utiliser chez soi : ainsi, il est possible de le télécharger et de l'installer sur son réseau local afin d'avoir son propre serveur.

IV. Cibles et acteurs

L'étude des cibles permet de ne pas se disperser lors de la mise en place des rubriques au sein du site et du design des nos logiciels. Notre projet reposant sur l'écoute de la musique, il concerne toutes les catégories d'âge et cela peu importe le sexe.

L'acteur de ce système sera l'Utilisateur. Il s'agit d'un particulier lambda qui souhaite écouter de la musique via son ordinateur ou son mobile. Il peut se connecter au serveur via le client, et peut donc rechercher et lire des musiques.

Certains Utilisateurs du système ont en plus le rôle d'Administrateur, ils peuvent gérer les morceaux, la mise en ligne du serveur ainsi que définir d'autres administrateurs. Cependant, ils auront également accès aux autres fonctionnalités du client (lire et rechercher des musiques…).

V. Difficultés

Le principal défi de ce projet était de réussir à créer un serveur sur Raspberry Pi. En effet, ses performances sont faibles (puissance et mémoire notamment). Ils nous a donc fallu optimiser toute la partie programmation et opter pour les solutions les plus adéquates.
Du côte client, les problèmes de puissance et de stockage sont moins importants. Cependant, nous avons tout de même fait en sorte que l'application soit rapide et fonctionnelle. Par contre, puisqu'elle a été réalisée sur différentes plateformes, des questions de compatibilité se sont alors posées.

PARTIE II : ARCHITECTURE ET CHOIX TECHNIQUES

I. Réflexions avant développement

Avant d'entamer la rédaction du cahier des charges, il semblait nécessaire de définir quels langages et quels outils seraient utilisés pendant le projet. Premièrement, un des objectifs initiaux était de rendre l'application multiplateforme : cette contrainte nous a poussé à réfléchir quant à l'utilisation d'un langage compatible multiplateforme et multi-OS. Pour répondre à ces attentes tout en utilisant les compétences acquises durant nos années de DUT, le Java était la meilleure solution.

Le logiciel de développement adapté à Java dont nous avions connaissances était Eclipse. C'est donc en toute logique que nous avons choisi de l'utiliser. Le partage de fichiers s'est fait via Dropbox, un logiciel simple à mettre en place et synchronisant toutes nos données automatiquement. De plus, en cas d'erreur, il nous permettait de récupérer une ancienne version de notre travail. En ce qui concerne la rédaction de documents en collaboration, Google Drive est apparu comme la plateforme adéquate. Finalement, le maquettage a été réalisé grâce au logiciel Adobe Photoshop.

II. Evolution du cahier des charges

Au fil du développement et en accord avec M. LOHIER, nous avons modifié certains éléments du cahier des charges. Ainsi, l'utilisation de LED s'est avérée superflue. En effet, si le serveur venait à s'arrêter, les LED seraient éteintes elles-aussi. De plus, le coût en ressources de cette fonctionnalité aurait été trop important comparé à son intérêt.
Lors de nos différents tests, la gestion des informations stockées sur le serveur s'est révélée être assez laborieuse sans connaissances en informatique. Ainsi, afin de rendre le serveur accessible et utilisable par tous, nous avons choisi de créer une interface d'administration. Celle-ci permettait initialement d'ajouter un compte utilisateur, d'ajouter un artiste, un album, une musique et par conséquent de les supprimer.
Suite à cette phase de développement, M. LOHIER nous a proposé d'ajouter une fonction d'import automatique. Cette dernière permet d'ajouter automatiquement à la base de données les musiques contenues dans un des disques durs reliés au serveur.
Du coté client, une fonction de répétition du morceau ou de la playlist a été rajoutée.

III. Développement

III.1. Organisation du projet

La dualité de l'architecture des dossiers du projet ("core" et "gui") est dûe à la séparation des fonctions permettant de communiquer avec le serveur et des fonctions d'affichage et d'interaction avec l'utilisateur.

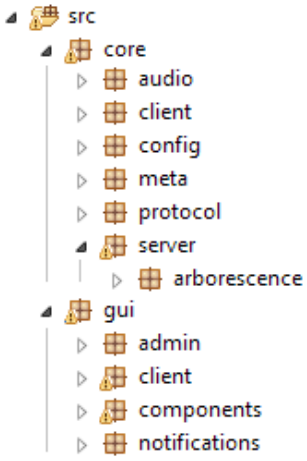


Figure : Organisation des dossiers du projet →

Le découpage du dossier “core” a été réalisé de façon à séparer les différents modules du coeur. Ainsi, la partie “audio” gère la lecture des musiques sur ordinateur. La partie “client”, elle, s’occupe de traiter les interactions entre l’interface graphique et le serveur. Les informations liées à la connexion au serveur (adresse, port etc.) et leur traitement se trouvent à l’intérieur du package “config”. Le dossier “meta” contient la structure des objets transmis à travers le réseau (musique, album…). La partie “protocole” gère la transformation des informations en bytes afin de réaliser la communication client-serveur. Finalement, la partie “server” se charge de lier client et base de données. Le sous-dossier “arborescence” contient les fonctions nécessaires au parcours d’un disque amovible.

Le “gui” est, lui, séparé de manière logique. En effet, les deux principaux dossiers “admin” et “client” correspondent respectivement à l’interface graphique d’administration et à l’interface graphique de l’utilisateur. Le package “components” contient les différents éléments graphiques nécessaires à la création des interfaces. Puisque les échanges client-serveur sont asynchrones, nous avons dû utiliser des gestionnaires d’événements définis dans le dossier “notifications”.

Android ne se comportant pas comme un PC, il a été nécessaire de revoir certains points de l’architecture du projet. Ainsi, le dossier “gui” a été réduit à un unique package de vues Android et même si le dossier “audio” du coeur existe toujours, c’est, cette fois, Android qui gère les fonctionnalités audio.

Le site web est le moyen de promouvoir et de télécharger le projet. De plus, sur celui-ci, les clients peuvent trouver des tutoriels d’aide d’installation et d’utilisation. Son développement ne sera pas plus développé car il n’est utilisé que pour l’acquisition du/des logiciel/s.

III.2. Détails de l’architecture

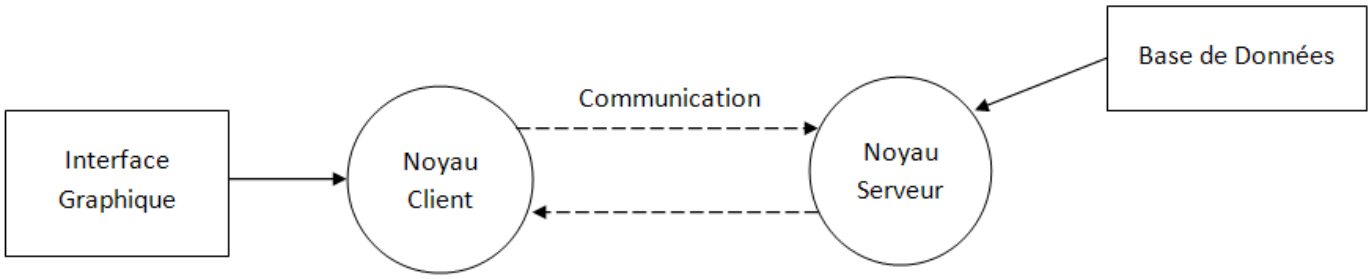
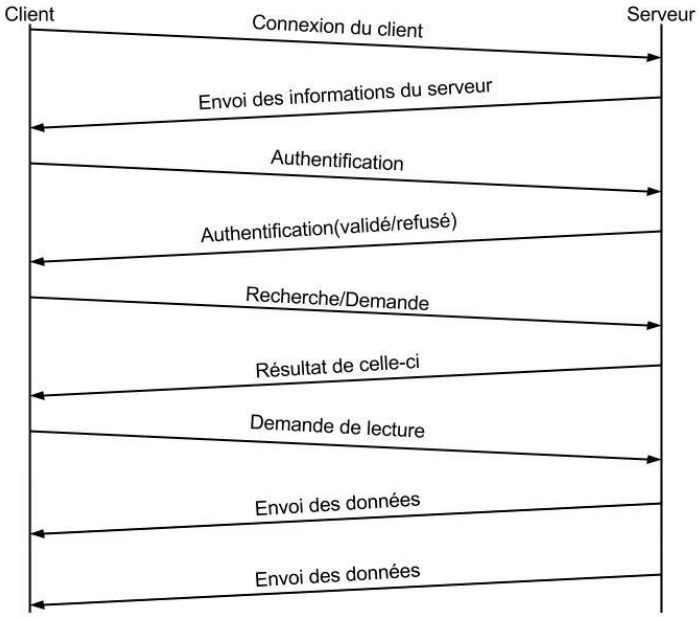


Figure 1 : Architecture simplifiée du projet

Le projet est basé sur une architecture client-serveur typique : deux noyaux s’échangeant des informations à travers le réseau et des interfaces permettant de communiquer avec l’extérieur.
Pour SpotiDut, nous avons choisi de développer une interface graphique reliée au client afin de simplifier les interactions avec le serveur. Une base de données est utilisée pour stocker les informations du serveur.

Pour plus de détails, des fiches techniques sont présentes en annexe.

III.3. Détails du protocole



La communication des deux noyaux à travers le réseau n'est possible que grâce à un protocole. A nouveau, le protocole créé pour le projet s'organise de la même manière que ceux déjà existants. On retrouve une phase de connexion, d'authentification puis d'échanges de données. Pour chaque type d'échange, des entêtes ont été développées. Ainsi, on retrouve les entêtes ADMIN, META, PLAY, DATA, AUTH correspondant chacune à un besoin fonctionnel.

Pour plus de détails, des fiches techniques sont présentes en annexe.

III.4. Détails du GUI

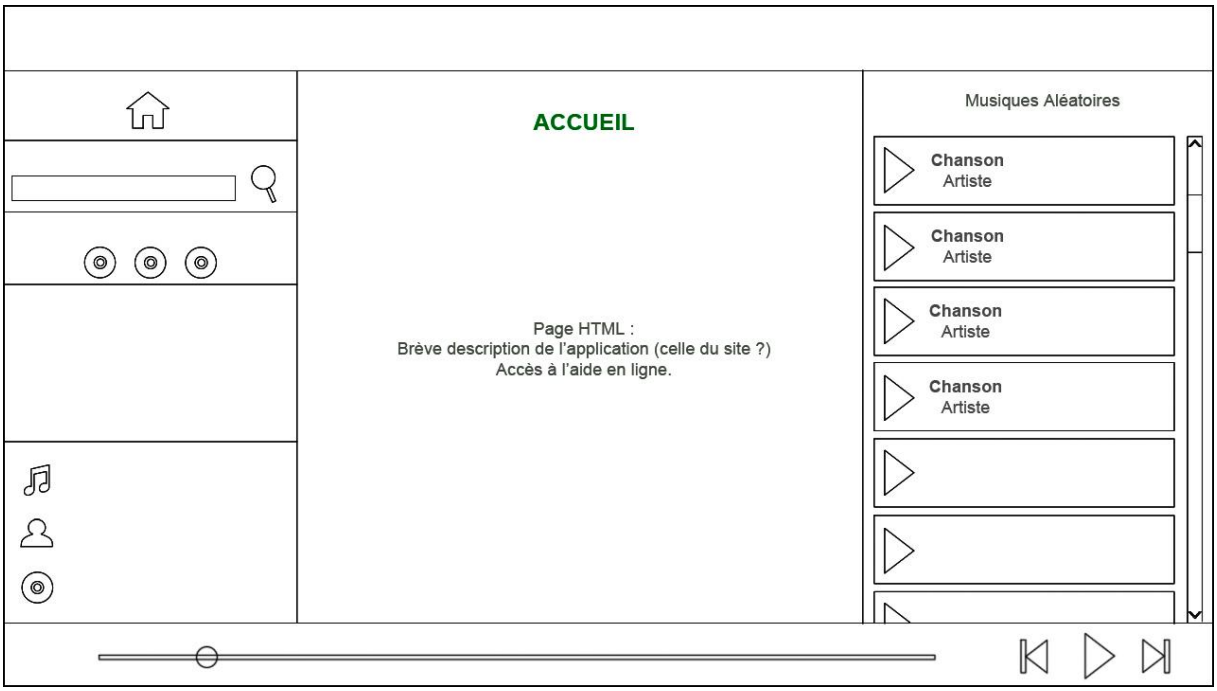


Figure 2 : Interface Graphique Cliente

Le maquettage de l'application du client a été réfléchi de manière à ce que son utilisation soit simple et intuitive. Ainsi, chaque fonctionnalité est représentée par un icône (bouton de lecture, de recherche…). L'interface peut être découpée en trois parties selon les fonctionnalités de chacune : le lecteur, la barre latérale et le panneau central (ici, accueil).

Le lecteur gère l'audio, c'est à dire la lecture, le déplacement dans la musique, le déplacement dans la liste de musiques et la répétition (absente sur cette maquette initiale).

La barre latérale a principalement un rôle informatif. Elle permet d'obtenir des informations sur la musique en cours de lecture ou sur la liste de musiques en cours. Elle permet également de faire des recherches ou de retourner à l'accueil.

Finalement, le panneau central est dynamique : il permet d'afficher l'accueil, les résultats de recherche ou la liste de musiques en cours de lecture.

IV. Difficultés techniques et solutions apportées

Comme dans tout projet, nous avons dû faire face à des problèmes techniques. Nous avons répertorié les plus dérangeants pour vous donner un aperçu de la complexité de notre projet.

Le premier point nous ayant posé problème a été le format MP3. En effet, ce format audio est un format propriétaire ce qui implique que Java n'a pas le droit de développer une solution permettant de manipuler un fichier de ce type. La solution à ce problème a été d'utiliser une librairie non standard de Java.

Nous nous sommes ensuite penchés sur le streaming. Deux solutions existantes sont ressorties de nos recherches. La première solution permettait de lire et de mettre en pause facilement un flux audio provenant du réseau, mais ne permettait pas de se déplacer dans l'audio reçu. La seconde solution permettait le déplacement dans le fichier audio, mais uniquement si celui-ci avait été totalement chargé au préalable.
Pour palier à cela, nous avons décidé de développer notre propre solution permettant de se déplacer dans la musique en cours de chargement.

Le troisième problème important à été Eduroam. En effet, le réseau Wi-Fi de l'université ne laisse pas passer toutes les connexions. Nous nous sommes donc retrouvés avec une Raspberry Pi impossible à configurer. Grâce a l'appui de M. LOHIER, nous avons pu ajouter l'adresse physique de la Raspberry Pi et ainsi l'utiliser sur le réseau.

La version Android de l'application nous a également posé beaucoup de soucis : les cours que nous avons eu ne nous permettant pas de développer une application aussi complexe.
L'import d'une librairie renvoyait une erreur lors du lancement de l'application qui se fermait aussitôt. C'est avec l'aide de M. FORAX que nous avons compris que l'erreur venait de la librairie audio que nous utilisions. Cette dernière était tout simplement incompatible avec Android qui gère cette partie nativement.

Suite à cette erreur, une nouvelle est apparue. L'authentification du client ne fonctionnait pas. C'est avec l'aide à M. PILLIET que la situation s'est débloquée.

V. Tests et Validation

Tout au long du projet, divers tests unitaires ont été réalisés afin de valider chacune des fonctionnalités développées. Ces tests nous ont permis de corriger et d'améliorer certains points afin de répondre au mieux aux attentes du cahier des charges.

Après ces tests unitaires, nous avons demandé à des utilisateurs lambda de tester notre application afin d'avoir un retour critique nous permettant de la faire évoluer.

Des tests de charges ont été réalisés. Le serveur sur Raspberry Pi supporte au maximum cinq clients demandant un stream de musiques. Cependant, nous n'avons pas réussi à déterminer si cette limitation était due à la capacité du réseau ou à la faible puissance de la Raspberry.

PARTIE 3 : GESTION DU PROJET

I. Communication

La communication au sein du projet est omniprésente et elle se distingue sous diverses formes:

- au sein du groupe :

Nous sommes un groupe composé de cinq étudiants et donc de cinq personnalités différentes. Malgré cela, le groupe est resté soudé et nous avons ressenti l'avancement du projet de la même manière, ce qui est essentiel dans la prise de décisions.

Chaque matin de projet, nous nous sommes souvent réunis pour discuter des objectifs à atteindre, des problèmes rencontrés et des tâches de chacun. Chaque membre a pu ainsi consulter le travail des autres et donner son avis afin d'améliorer le projet. Cette organisation nous a permis d'avancer rapidement tout en gardant un œil sur l'ensemble du travail à réaliser.

Malgré les quelques difficultés que nous avons connues, nous avons su être réactifs et nous avons réussi à nous mettre en accord pour essayer de les résoudre.

- avec le tuteur

Notre tuteur, M. LOHIER, a joué un rôle prépondérant dans le projet car il était présent pour nous conseiller, nous orienter et valider nos choix.

Le rapport entre l'équipe et M. LOHIER s'est avéré professionnel, dans la mesure où le projet avançait dans les temps et que nous le tenions régulièrement au courant de l'évolution de celui-ci.
En effet, nous avions des réunions avec lui, dans la mesure du possible, tous les mois du début jusqu'à la fin du projet. Il nous donnait son avis en fonction de l'avancement du projet, et nous posait des questions sur notre méthode de travail et sur l'organisation au sein du groupe. Évidemment, en tant que client, M. LOHIER est resté force de proposition.

II. Organisation et répartition des tâches

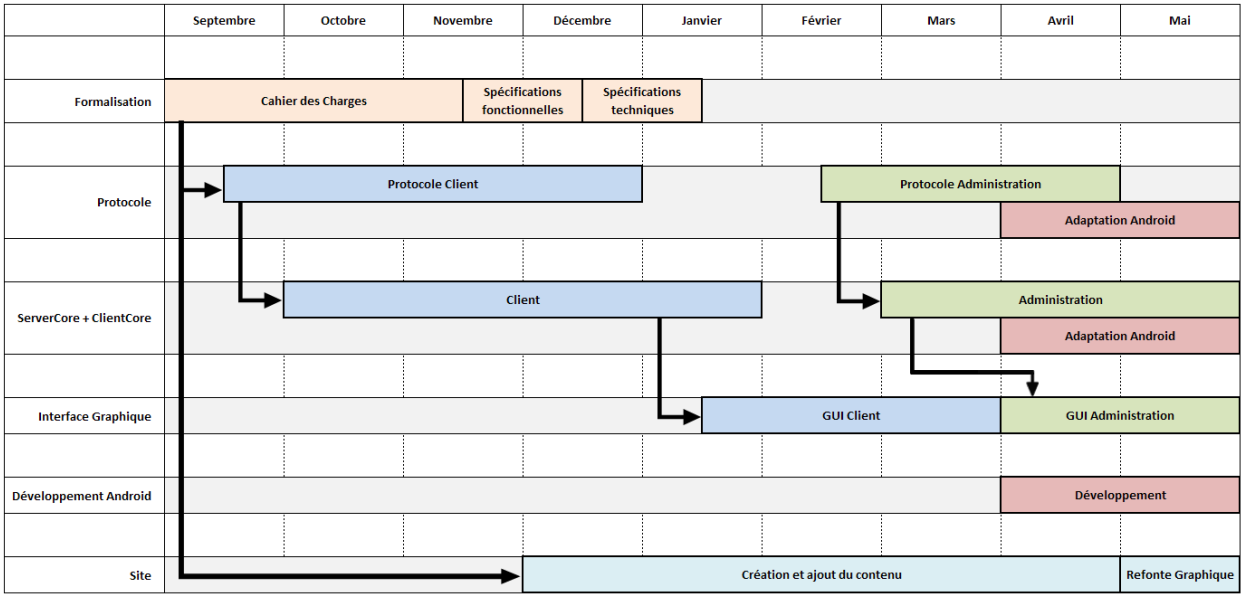


Figure 3 : GANTT du projet (Voir annexe)

Afin de mener à bien le projet et suite à sa diversité et son ampleur, il a été nécessaire de répartir les tâches. Ces dernières ont été attribuées en fonction de nos compétences et de nos préférences de manière à ce que chacun puisse apporter sa pierre à l'édifice. Néanmoins, comme expliqué précédemment, tout le travail a été réfléchi en groupe et les avis de chaque membre ont été pris en considération.

Ainsi, après définition des spécifications fonctionnelles et techniques (Octobre à Février), l'équipe constituée de cinq personnes a été découpée en deux groupes :

- le premier groupe a travaillé à la spécification précise du protocole de communication client/serveur, et l'élaboration de tests techniques en rapport avec le réseau et les supports audio. Ce groupe était composé de Thibault, Pierre et Baptiste.

Note : Pierre a ensuite commencé la conception du site web, une fois la spécification du protocole terminée, laissant Thibault et Baptiste s'occuper du développement pur.

- le second groupe a travaillé à la recherche concernant la configuration de la Raspberry, à la spécification du client pour pc et plus particulièrement aux interfaces graphiques. Ce groupe était composé de Axel et Vanessa.

Le but était d'avancer au maximum ce qui pouvait l'être, tout en nous formant sur les nouveaux domaines qui n'avaient pas été vus en cours mais étaient nécessaires.

Durant la période de développement dite "lourde", des réunions régulières et une planification en amont ont permis de développer l'interface graphique et la partie communication en parallèle, les deux devant se compléter parfaitement à la fin.

Note : Les phases de recherche ne sont pas indiquées dans le Gantt.

Suite à la première phase, une seconde et petite phase (début Février) nous a permis de rejoindre les deux développements (chacun ayant fait évoluer l'autre avec l'apparition de nouveaux besoins et de corrections au fil des réunions de la première phase). Un exemple de ces regroupements est la mise en place d'un système nommé les "notifications listeners" (pour plus d'informations, voir l'annexe "Fiche Technique : Les NotificationListeners").

La dernière phase (Février à Juin) a suivi un développement agile : le projet étant bien avancé, nous avons accepté plusieurs ajouts au cahier des charges (dont certains optionnels), puisque le temps restant nous le permettait. Cette phase s'est organisée autour de mini-réunions chaque matin pour se recentrer et s'assurer que nous ne dévions pas de l'objectif.

Nous avons choisi de venir à l'IUT presque chaque jour durant toutes les semaines de projet, pour conserver un rythme aussi régulier que possible, pouvoir nous réunir facilement et donc pour mieux prévoir l'avancement du projet et l'approche des deadlines. Cette organisation a grandement simplifié la communication et nous a beaucoup aidé.

III. Difficultés rencontrées dans l'organisation et la communication

Comme dit précédemment, nous étions tous les jours à l'IUT. Cependant, chacun de nous avait ses demandes (compétitions sportives, entretiens…). Ces dernières, étant imprévues, ont légèrement ralentis l'avancement du projet.

La répartition des tâches a été faite au mieux pour convenir à chacun sans ralentir le projet. Néanmoins, il était parfois difficile d'attribuer des tâches à chaque membre du groupe, certaines missions nécessitant des compétences particulières.

PARTIE IV : BILAN

I. Conclusion du projet

M. LOHIER a validé notre projet. Les fonctionnalités initiales ont été développées (serveur + client Mac, Windows, Linux) et de nouvelles fonctionnalités ont vu le jour (administration, import automatique notamment). Les tests réalisés nous ont permis de vérifier le bon fonctionnement de notre travail.

Nous apportons un début de solution à l'application Android : il est possible de se connecter et de rechercher des musiques. Cependant, le streaming de ces dernières n'est pas encore fonctionnel par manque de temps et de compétences. Un mois de travail aurait été largement suffisant pour implémenter cette dernière fonctionnalité.

Une documentation du code et du fonctionnement du projet a été réalisée afin de permettre à quiconque le souhaite de reprendre le projet.

II. Conclusion humaine

Au lancement du projet, nos compétences en Java audio et graphique, ainsi qu'en Android, étaient minimes. En effet, nous n'avions jamais manipulé l'audio, notre connaissance de la librairie d'interface graphique Java était limitée à la création d'une calculatrice basique et nos cours d'Android n'avaient pas encore débuté. Ces lacunes nous ont appris à être autodictacte : c'est à dire à chercher des informations et des explications ainsi qu'à apprendre par nous même. Des phases de recherche intenses ont été nécessaires tout au long du projet et cet entraînement ne peut être que bénéfique pour notre apprentissage.

Chacun des groupes formé a développé des compétences spécifiques à son travail. Par exemple, Thibault, Pierre et Baptiste ont appris le fonctionnement de l'audio sous Java. Axel et Vanessa, quant à eux, ont appris à maitriser les librairies graphiques de Java.

Nous pensons maintenant que nous sommes plus apte à travailler en équipe et que ce projet nous a permis de tirer des enseignements solides et une expérience forte, car nous avons pu voir concrètement le travail nécessaire à la conduite d'un projet, de l'élaboration du cahier des charges jusqu'au produit fini à remettre au client.

Au final, cette année se révèle sans aucun doute réellement bénéfique pour l'avenir car nous participerons certainement à d'autres projets voir à leur conduite.

GLOSSAIRE

Base De Données: Une base de données ou banque de données est un outil permettant de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité.

Entête : Ce sont les premiers caractères lus dans un message. Ils permettent son authentification et la définition de son traitement.

Ethernet : Norme de protocole de réseau très répandue. La norme Ethernet utilise un type de connection filaire. C'est la norme la plus utilisée pour les réseaux locaux et la connexion internet en filaire.

Java : Technologie composée d'un langage de programmation orienté objet et d'un environnement d'exécution.

Plateforme : Ensemble constitué d'un système d'exploitation et d'un ordinateur. Il s'agit donc d'une architecture matérielle qui permet à l'utilisateur d'effectuer un grand nombre de tâches.

Protocole: Moyen de communication entre deux machines, généralement un client et un serveur.

Raspberry Pi : Mini ordinateur mono-carte à processeur ARM de la taille d'une carte de crédit.

Serveur : Un serveur informatique est un dispositif matériel ou logiciel qui offre des services à différents clients. Quelques services utilisés ici :

- le stockage en base de données ;
- la gestion de l'authentification.

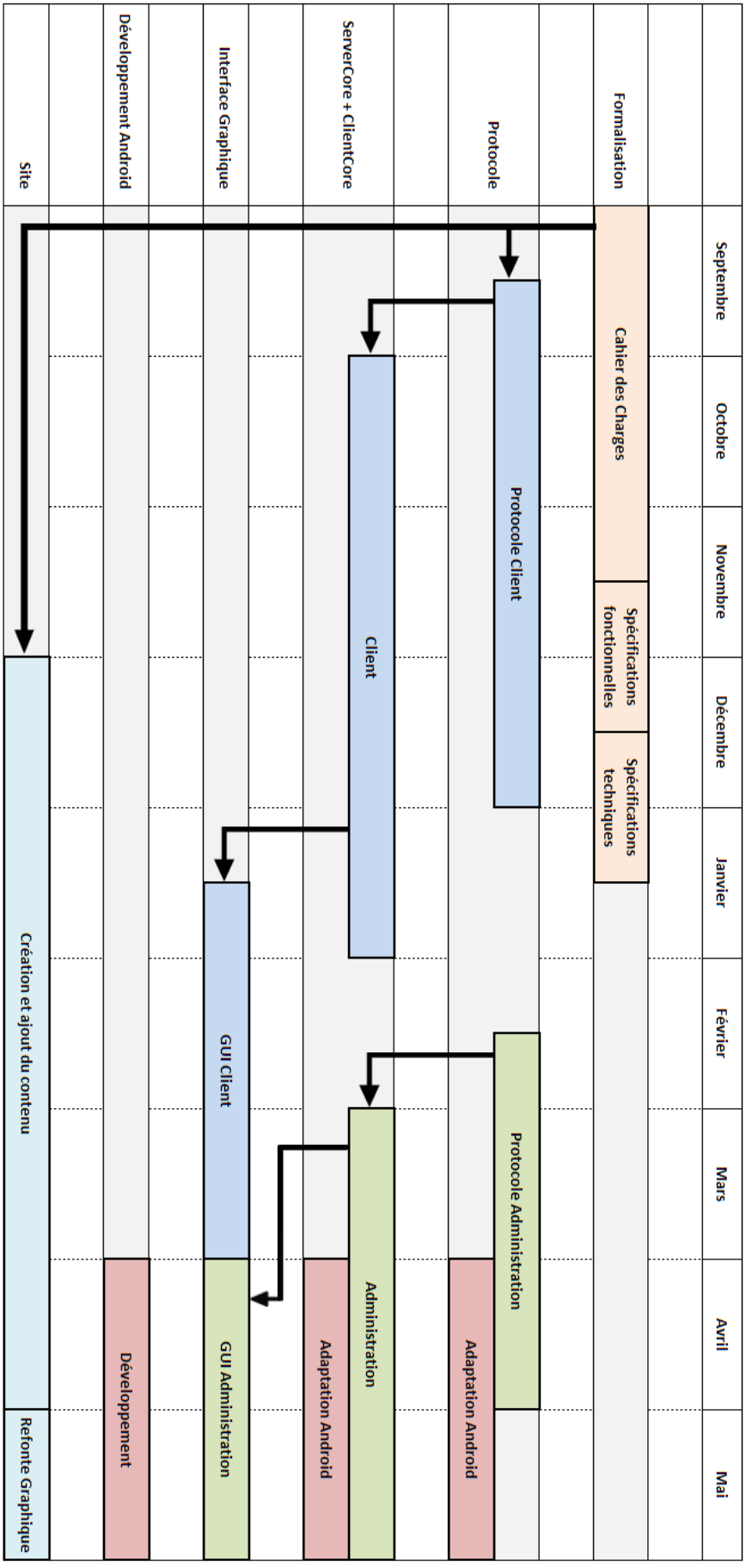
Figure 4 : Raspberry Pi

Streaming : Désigne l'envoi d'un contenu en "direct". On envoie ici, depuis le serveur, une musique que le client ne possède pas forcément sur son ordinateur.

Wi-Fi : Technologie permettant de se connecter à Internet sans utiliser de câble.

ANNEXES

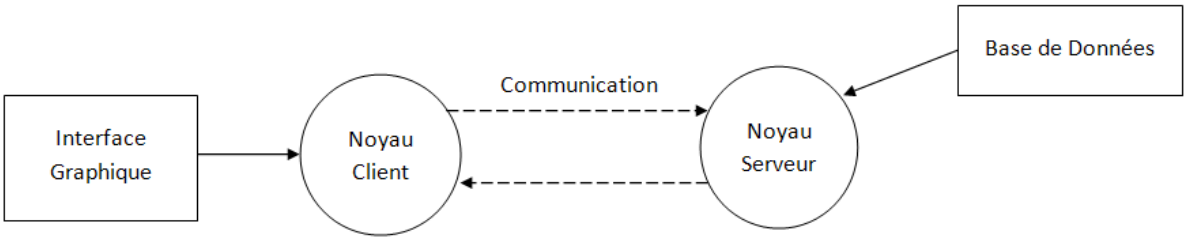
Figure 3 : GANTT du projet



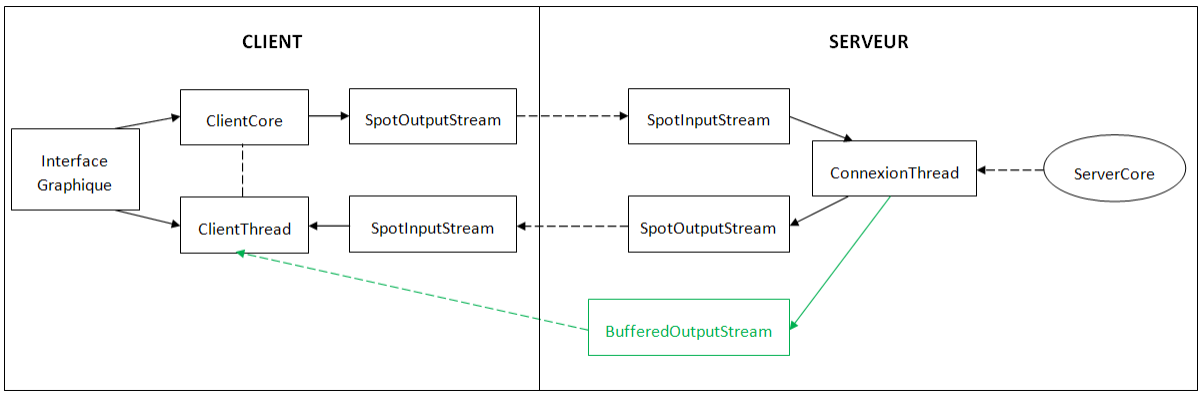
Fiche technique : Architecture Client-Serveur

Le projet est basé sur une architecture client-serveur typique : deux noyaux s'échangeant des informations à travers le réseau et des interfaces permettant de communiquer avec l'extérieur.
Ici, l'interface graphique est utilisée pour communiquer avec le serveur en passant par le coeur du client.

En ce qui concerne le dialogue entre le serveur et la base de données, nous aurions pu créer une interface. Cependant, nous avons choisi d'utiliser directement le coeur du serveur pour leur communication.



Dans le schéma ci-dessous, nous avons détaillé un peu plus la structure.



Le serveur :

Nous avons une partie serveur composée d'un coeur, de threads et de flux entrant et sortant.

Le coeur du serveur (ServerCore) est présent afin d'accueillir les différents clients, ordinateurs ou Android, et attribue à chacun un thread (ConnexionThread). Celui-ci sert à gérer les différentes requêtes du client qui y sont associées. Il s'occupe ainsi d'authentifier le client, de faire les recherches en bases de données et de renvoyer les résultats, de streamer la musique demandée et, pour un administrateur, de permettre l'ajout de ressources au serveur.

Pour communiquer avec la partie client, le thread du serveur gère un flux entrant (SpotInputStream) chargés de recevoir les différentes requêtes du client et un flux sortant (SpotOutputStream) chargé, lui, de répondre au client.

Le client :

La partie client est composée d'un coeur, d'un thread, de flux entrant et sortant et d'interfaces graphiques.

Elle possède deux interfaces : une pour la lecture de musique et une deuxième pour l'administration du serveur. Ces deux interfaces communiquent directement avec le coeur client réalisant les demandes au serveur et reçoivent les réponse grâce à des NotificationsListener. Ces derniers sont le moyen de renvoyer les informations à l'interface graphique sans que celle-ci ne soit bloquée lors de l'attente de la réponse. En effet, le client envoie sa requête puis lance un NotificationListener. Lorsque la réponse du serveur arrive, le NotificationListener est "activé" et la méthode qui lui est attribuée (comme l'affichage des résultats d'une requête) s'éxecute.

Le coeur du client (ClientCore) permet au client de faire des demandes au serveur : authentification, recherches d'informations, lecture d'une musique ou encore ajout de ressources au serveur. Le thread (ClientThread) sert à récupérer les réponses du serveur. Il est nécessaire puisque la connexion est asynchrone. De plus, ce thread sert à gérer l'obtention des données de lecture sans bloquer l'application. Etant donné que le thread est la partie du client qui s'occupe de la réception des données, celui-ci est lié au flux entrant (SpotInputStream). Logiquement, puisque l'envoi des informations est géré par le coeur, c'est celui-ci qui est lié au flux sortant (SpotOutputStream).

Fiche technique : Protocole côté administration

Le logiciel administrateur utilise trois entêtes différentes : AUTH, pour pouvoir se connecter ; META, pour obtenir les informations nécessaire à la gestion (la liste des artistes pour la création d'albums, la liste des utilisateurs pour pouvoir les supprimer, par exemple) et ADMIN, indiquant au serveur d'ajouter une ressource (un utilisateur, une musique, un album ou un artiste).

Pour les requêtes avec une entête AUTH le serveur répond avec 2 booleans qui permettent de savoir si le client existe et s'il a les droits d'administration. Les requêtes avec l'entête META lui renvoient la liste d'éléments voulus et, en ce qui concerne les entêtes ADMIN, une réponse de validation ou d'erreur est envoyée.

Tableau requête-réponse :

AUTH

Requête			Réponse
Entête *(1 byte)*	Données *(1 int longueur + 1 byte par caractère)*	Données *(1 int longueur + 1 byte par caractère)*	Boolean *(1 byte accès + éventuellement 1 byte admin)*
AUTH	longueur du login + login	longueur du mot de passe + mot de passe crypté	accès ou non et admin ou pas

META

Requête					Réponse
Entête *(1 byte)*	MetaType *(1 byte)*	Composant graphique destination *(1 int)*	Trié ou aléatoire *(1 byte)*	Mot clé de la recherche (ou id)	ResultList
META	MUSIC	target	ResultListOrder	keyword	Liste de musiques
	ARTIST				Liste d'artistes
	ALBUM				Liste d'albums
	ALBUM_ID		id		Liste de musiques dans l'album avec l'id
	ARTIST_ID				Liste d'albums de l'artiste avec l'id

ADMIN

<table>
<tr><th colspan="4">Requête</th><th>Réponse</th></tr>
<tr><th>Entête
(1 byte)</th><th>AdminCommand
(1 byte)</th><th>MetaType
(1 byte)</th><th>Données</th><th>AdminAnswers</th></tr>
<tr><td rowspan="8">ADMIN</td><td>ADD_USER</td><td colspan="2">user</td><td rowspan="8">OK
NO_SUCH_RESSOURCE
ALREADY_EXISTING_RESSOURCE
NON_ADMIN_SESSION
META_ERROR</td></tr>
<tr><td>DELETE_USER</td><td colspan="2">id</td></tr>
<tr><td rowspan="3">ADD_RESSOURCE</td><td>MUSIC</td><td>AdminMusic</td></tr>
<tr><td>ARTIST</td><td>Artist</td></tr>
<tr><td>ALBUM</td><td>Album</td></tr>
<tr><td rowspan="3">DELETE_RESSOURCE</td><td>MUSIC</td><td rowspan="3">id</td></tr>
<tr><td>ARTIST</td></tr>
<tr><td>ALBUM</td></tr>
</table>

Fiche technique : Protocole côté client

Le logiciel client est la partie lourde : c'est pourquoi il nécessite plus d'entêtes que le logiciel administrateur. Celui-ci utilise donc les entêtes AUTH, pour pouvoir se connecter, META, pour avoir les informations afin de pouvoir lire les musiques, DATA, pour le tranport des données audio et PLAY, pour demander la lecture d'une musique.

Pour les requêtes avec une entête AUTH, le serveur répond avec 2 booleans permettant de savoir si le client existe. Pour celles avec l'entête META, il renvoie la liste d'éléments voulus (musique, artiste, album). Enfin, pour les requêtes avec l'entête PLAY, le serveur renvoie l'audio format concernant la musique puis les bytes de la musique.

Tableau requête-réponse :

AUTH

Requête			Réponse
Entête	Données	Données	Boolean
AUTH	longueur du login + login	longueur du mot de passe + mot de passe crypté	accès ou non et admin ou pas

META

Requête					Réponse
Entête	MetaType	Composant graphique destination	Trié ou aléatoire	Mot clé de la recherche (ou id)	ResultList
META	MUSIC	target	ResultListOrder	longueur du keyword + keyword	Liste de musiques
	ARTIST				Liste d'artistes
	ALBUM				Liste d'albums
	ALBUM_ID		id		Liste de musiques dans l'album avec l'id
	ARTIST_ID				Liste d'albums de l'artiste avec l'id

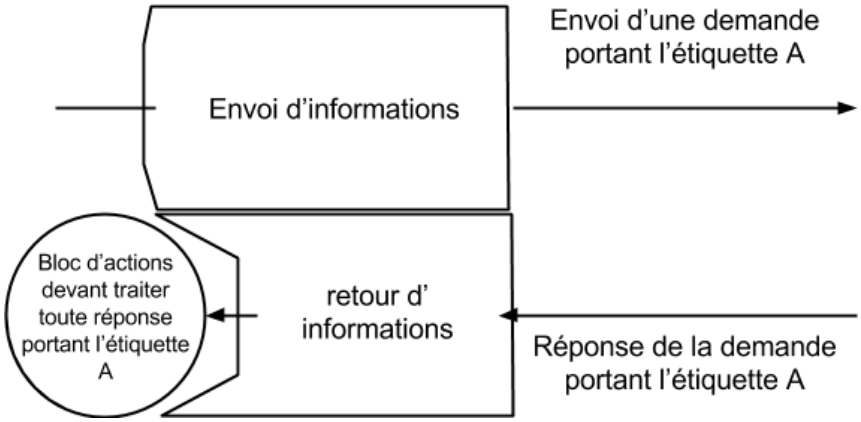
PLAY-DATA

<table>
<tr><td colspan="2">Requête</td><td colspan="4">Réponse</td></tr>
<tr><td>Entête
(1 byte)</td><td>Id de la musique voulue
(1 int)</td><td>Entête
(1 byte)</td><td>DataType
(1 byte)</td><td></td><td></td></tr>
<tr><td rowspan="2">PLAY</td><td rowspan="2">nombre</td><td rowspan="2">DATA</td><td>AUDIOFORMAT</td><td>Longueur de la musique</td><td>Information sur la musique</td></tr>
<tr><td>MUSIC</td><td>Longueur du tableau de bytes</td><td>Tableau de bytes</td></tr>
</table>

Fiche technique : Les NotificationListeners

Les NotificationListeners sont un système permettant à l'interface de savoir quand le coeur a fini de traiter une de ses demandes (une recherche de musiques, par exemple).

Schéma :



Explications :

L'interface possède par exemple une partie dédiée à la recherche de musiques. La réponse doit être affichée dans un panneau. La portion d'instructions devant s'occuper de traiter la réponse est donnée au coeur, comme "greffée", et y est reliée de manière à ce que le coeur sache à quelle étiquette ce bloc correspond.

L'interface peut ensuite demander une recherche, en lui donnant une étiquette; la réponse sera récupérée lorsqu'elle reviendra, puis traitée par le bloc d'instructions préalablement donné par l'interface. Si une réponse possède une étiquette ne correspondant à aucun bloc de traitement, alors celle-ci est oubliée.

Ce système permet l'envoi de plusieurs requêtes en parallèle, chacune pouvant revenir dans n'importe quel ordre; il trouve son utilité dans divers cas : lorsque l'application doit continuer de lancer diverses recherches tout en laissant l'utilisateur faire les siennes, ou qu'une interface a plusieurs recherches à lancer en même temps pour obtenir des informations. Comme nous ne savons pas quand reviendront les réponses, le coeur utilise ce système pour savoir qui doit traiter les réponses qu'il reçoit.