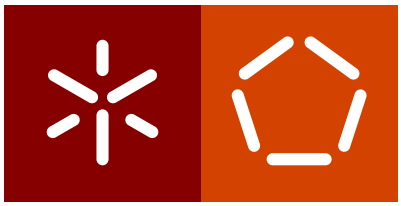
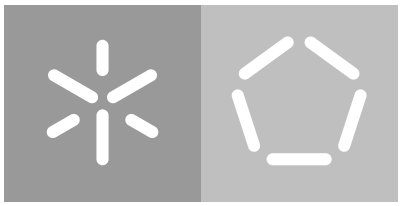


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jorge Miguel da Silva Oliveira

Serviço de Acesso Remoto a Coleções Musicais em Servidores Privados

31 de Julho de 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jorge Miguel da Silva Oliveira

Serviço de Acesso Remoto a Coleções Musicais em Servidores Privados

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Bruno Alexandre Fernandes Dias

31 de Julho de 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS

Em primeiro lugar quero deixar um agradecimento especial ao professor Bruno Dias, orientador da dissertação, pela sugestão do tema e por se mostrar disponível para me ajudar e encaminhar durante este longo percurso. A sua dedicação e entusiasmo pelo tema impulsionaram-me, conseguindo, assim, manter o foco e determinação. Com toda a sua experiência guiou-me em todas as situações, inclusive as mais complicadas, fazendo com que eu escolhesse o melhor caminho para resolver os problemas.

Quero ainda agradecer a toda a minha família pelo constante apoio durante todo o meu processo acadêmico, e em especial, na escrita desta dissertação.

Por fim, agradeço ainda a todos os meus amigos por me proporcionarem vários momentos de descontração e convívio essenciais para estar revigorado.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

ABSTRACT

Streaming content has completely changed the way we consume it. The arising of the Internet, and streaming services, has allowed people to access a big collection of music without buying it, just by subscribing to a service or to use it for free with the consumption of publicity.

But, and for those who have a local collection? They can't use these platforms unless the service allows the user to buy the content, store it in a local environment, play it there, and the user could play it remote by streaming.

The use of clouds appears to be a good solution for this group of people. Unfortenelly, when the collection is really big, the additional costs of storage of the cloud could be a big problem. In addition, normally, the user interface of these services isn't enjoyable, i.e., isn't user-friendly.

So, a hybrid system could be the ideal, a streaming service with cloud services that allow storing the private collection. This is the best for the user but will force them to store all the collection or at minimum a part of the collection which they want to access, in an external source, with a strict organizational structure. For a new user, this couldn't be a problem but is for a user who already has a big collection.

This dissertation has proposed a remote access service to these private collections so that with just a simple gadget (smartphone, pc, ...) the user can access the content stored on their private server anywhere on the planet, without change the original location of the collection. This service was built using only normalized and open-source technologies.

Based on the proposed architecture, was also developed a prototype with the main function of the system, like the possibility to play music from the private collection on a smartphone Android. The conclusion of the tests made to the prototype was that this alternative solution could be very good for the people who want that their collection remains in one place, and, at the same time, can play it remotely.

Keywords: Mobile Application; Music; Remote Server; Streaming.

RESUMO

A transferência via *streaming* mudou completamente a forma como consumimos conteúdo audiovisual. O aparecimento de plataformas de *streaming* de áudio revolucionou o mundo da música, permitindo que um utilizador possa ter acesso a uma panóplia de conteúdo sem ter de adquirir os direitos dos mesmos, apenas subscrevendo o serviço, pagando uma mensalidade ou gratuitamente em troca do consumo obrigatório de publicidade.

No entanto, este tipo de serviço não é útil a quem possui uma coleção musical privada armazenada localmente. Neste caso não se pode recorrer a uma destas plataformas de *streaming* a não ser nos casos pouco comuns em que a música foi adquirida num serviço de *streaming* que permita a compra desse conteúdo para armazenamento e reprodução local e, em simultâneo, a sua reprodução remota através desse mesmo serviço de *streaming*.

Uma alternativa possível para reprodução remota duma coleção privada é o uso de *clouds* genéricas para o seu armazenamento, mas esta solução não é prática para coleções musicais de maior tamanho, pode acarretar custos adicionais no uso do serviço de *cloud*, a interface dos reprodutores musicais dos serviços de *cloud* são relativamente pobres e as exigências de qualidade de serviço podem ser maiores do que as que seriam necessárias apenas para *streaming* de música.

Existem também soluções híbridas, ou seja, serviços de *streaming* de música que são também serviços de *cloud* para coleções musicais privadas. Esta tipo de solução mais recente é a que oferece melhores funcionalidades, mas obriga a que os utilizadores transfiram todos os ficheiros e mantenham a gestão da sua coleção completa duma forma remota. Esta característica pode não ser relevante para novos utilizadores que estão a começar a sua coleção musical, ou a sua coleção é relativamente pequena, mas pode ser uma desvantagem importante para utilizadores que já possuam uma coleção grande, gerida duma forma local com ficheiros de diversos formatos.

Nesta dissertação foi proposto um serviço de *streaming* de acesso remoto a coleções privadas, para que qualquer utilizador com as devidas permissões de acesso, possa reproduzir o conteúdo da sua coleção em qualquer lugar e em qualquer dispositivo fixo ou móvel (desde que esteja assegurada uma ligação à rede Internet com uma qualidade adequada para *streaming* de áudio), sem que seja necessário transportar ou transferir a coleção do seu armazenamento local. Este serviço foi definido utilizando apenas tecnologias normalizadas e *open-source*.

Baseado na arquitetura do sistema proposto, foi desenvolvido um protótipo implementando as principais funcionalidades do sistema, incluindo a reprodução de música duma

coleção privada num *smartphone* com o sistema operativo Android. Os resultados dos testes feitos com o protótipo permitiu concluir que a solução proposta pode constituir uma alternativa viável aos serviços já existentes, sobretudo quando o utilizador prefere não transferir (e manter atualizada) a sua coleção privada para um serviço na rede.

Palavras Chave: Aplicação Móvel; Música; Servidor Privado; *Streaming*.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

A

ANS1 Abstract Syntax Notation One.

API Application Programming Interface.

D

DHCP Dynamic Host Configuration Protocol.

DLNA Digital Living Network Alliance.

F

FTP File Transfer Protocol.

H

HLS HTML Live Stream.

HTTP Hypertext Transfer Protocol.

J

JSON JavaScript Object Notation.

JWT Json Web Token.

M

MP3 MPEG-1 Audio Layer 3.

P

PCM Modulação por Código de Pulso.

R

RTCP Real-time Cotrol Protocol.

RTP Real-time Transport Protocol.

S

SQL Structured Query Language.

SSDP Simple Service Discover Protocol.

T

TCP Transmission Control Protocol.

TTL Time to Live.

U

UPNP Universal Plug and Play.

URL Uniform Resource Locator.

1

INTRODUÇÃO

Outrora, o conteúdo multimédia na rede *Internet* era acedido através do mecanismo de *download*, i.e., a transmissão de conteúdo digital (áudio, vídeo, imagem, texto, etc.) era feito a partir de um dispositivo-fonte, normalmente um servidor remoto, para um cliente (um computador local, *smartphone*, etc.). Esta transmissão era despoletada pelo cliente que enviava um pedido para o servidor, requisitando-lhe um conteúdo específico que era posteriormente entregue ao cliente, caso este tivesse permissão para aceder aos ficheiros digitais requisitados. A reprodução desse conteúdo era depois responsabilidade do utilizador e era feito no mesmo dispositivo onde os ficheiros transferidos eram armazenados. Para que o utilizador pudesse reproduzir o conteúdo tinha de aguardar que a transferência estivesse totalmente completa, o que podia ser um processo demorado. Além disso, fora do dispositivo local o utilizador não podia ter acesso a esses ficheiros a não ser que fizesse mais cópias para outros dispositivos móveis.

Por outro lado, com o aparecimento de tecnologia que permitia retirar o conteúdo áudio dos CDs (ou o conteúdo multimédia dum DVD) e fazer cópias para unidades de armazenamento local para posterior reprodução em sistemas computacionais fixos ou móveis, passou a ser possível aos utilizadores domésticos construir e gerir coleções musicais utilizando sistemas locais de armazenamento, ligados diretamente ou em rede local, aos seus sistemas computacionais pessoais. Através de aplicações multimédia a correr nos sistemas computacionais que têm acesso aos dispositivos de armazenamento, pode depois reproduzir-se o conteúdo destas coleções musicais em formato digital.

A certa altura, devido à evolução tecnológica dos dispositivos computacionais de armazenamento local, dos mecanismos e aplicações de tratamento do conteúdo multimédia e das capacidades das tecnologias de implementação de redes locais IP, passou a ser possível endereçar independentemente o problema do armazenamento e organização das coleções musicais (em que o conteúdo tanto pode provir de cópias diretas de CDs ou DVDs como de ficheiros obtidos por *download* na Internet) e o problema da sua reprodução nos dispositivos ligados à rede local mantida pelo utilizador doméstico. Com o aparecimento de protocolos de partilha de conteúdos multimédia entre dispositivos ligados em redes locais (DLNA, UPnP, DAAP, etc.) passou a ser possível o acesso não direto a uma coleção musical por vários dispositivos em simultâneo na mesma rede local, i.e., uma espécie de acesso

remoto por *Streaming*, mas em contexto de rede local. Com a crescente implementação modular dos dispositivos que implementam os vários componentes dum sistema completo de armazenamento, gestão e reprodução de coleções musicais numa rede local, é possível construir sistemas híbridos em que as fontes de informação multimédia são completamente digitais, a gestão e controlo da reprodução musical é feito por aplicações multimédia a correr em dispositivos computacionais fixos ou móveis e os componentes responsáveis pela reprodução áudio final tanto podem ser equipamentos tradicionais puramente analógicos como equipamentos de reprodução digitais (como, por exemplo, auscultadores Bluetooth sem fios).

O rápido desenvolvimento das tecnologias de acesso à Internet, nomeadamente as tecnologias de acesso sem fios por *Wi-Fi* ou por redes celulares, tornou possível o aparecimento de serviços de *streaming* por pedido (também designados por sistemas de *streaming offline* ou *streaming on-demand*) de conteúdos multimédia na Internet.

1.1 *streaming* de áudio

Os sistemas clássicos de *streaming* multimédia, e em particular de *streaming* áudio, utilizam um protocolo de comunicação aplicacional que transfere o conteúdo digital dum ficheiro multimédia entre o dispositivo onde o conteúdo está armazenado (ou servidor de *streaming*) e um dispositivo que o recebe e reproduz (ou cliente de *streaming*). O dispositivo que faz a reprodução não armazena localmente o conteúdo recebido, ainda que possa existir um mecanismo de cache que armazena parte ou a totalidade do ficheiro durante um curto espaço de tempo.

Normalmente a aplicação de reprodução de conteúdo áudio do cliente costuma incluir funcionalidades de procura (procurar uma ou mais músicas por título, intérprete, álbum, obra, tipo, etc.) e de controlo da reprodução (iniciar reprodução, pausar a reprodução, avançar/recuar rapidamente, saltar trechos, etc.). Adicionalmente, é possível que os clientes também implementem funcionalidades mais avançadas como gestão de listas de reprodução, de filtragem e/ou recomendações de conteúdo dependendo do tipo de conteúdo normalmente acedido, de compra definitiva de conteúdo, manutenção de estatísticas, etc.

Uma componente importante dum sistema de *streaming* de áudio é a tecnologia de comunicação entre o servidor e o cliente. Esta tecnologia inclui um protocolo aplicacional que é implementado sobre um (ou vários) protocolo(s) de transporte. Como a reprodução de conteúdo áudio por pedido não é muito exigente em termos de atraso, não é costume utilizar-se protocolos de *streaming* específicos normalizados, como o Real-time Transfer Protocol (RTP) [1] ou o Real-time Streaming Protocol (RTSP) [2]. Estes protocolos são usados sobretudo em sistemas de *streaming* áudio ou vídeo em que a qualidade de serviço exigida é ditada sobretudo por baixos valores de latência, sincronia entre dados e imunidade à perda

de uma percentagem não irrelevante de pacotes/dados (por exemplo, para sistemas de videoconferência, videochamada, audioconferência ou telefonia). No caso de sistemas de *streaming* de áudio para reprodução de coleções musicais as maiores exigências, em termos de qualidade de serviço, centram-se nos seguintes parâmetros:

RITMO DA TRANSFERÊNCIA DE DADOS DISPONÍVEL (OU LARGURA DE BANDA FIM-A-FIM) - de forma a adaptar o ritmo disponível ao *streaming* áudio o protocolo aplicacional, em geral, utiliza mecanismos de codificação que permitem a compressão dos dados do conteúdo original (como está armazenado no servidor) por forma a que o atraso seja minimizado e permita uma experiência adequada ao utilizador; os mecanismos de codificação podem utilizar compressão sem perdas de informação (relativamente à versão original armazenada no servidor) ou, quando a largura de banda é mais limitada, os mecanismos de codificação podem implementar compressão com perda de informação; no cliente destino os dados são descodificados antes da reprodução; este processo de descodificação no cliente pode ser realizado num único passo, i.e., os dados são descodificados diretamente para o módulo de reprodução musical analógica (nestes casos a codificação inicial é feita para um formato normalizado como MP3, AAC, etc., e a qualidade áudio é determinada pelo formato de codificação), ou o processo de descodificação no cliente pode ser realizado em dois passos, i.e., primeiro os dados transferidos são descodificados para o formato original em que estavam armazenados e só depois são entregues ao módulo de reprodução musical analógica (nestes casos a qualidade áudio é determinada pelo formato em que o conteúdo está disponível no servidor – quando a transferência utiliza um mecanismo de compressão sem perda de informação –, ou pelo melhor formato possível na altura da descodificação – quando a transferência utiliza um mecanismo de compressão com perda de informação; de notar que a melhoria significativa no ritmo de transferência fim-a-fim disponível globalmente na Internet tem vindo a permitir que o acesso remoto a conteúdo musical por sistemas de *streaming* áudio possa ser feito com uma qualidade cada vez melhor;

FIABILIDADE DA REDE - para sistemas de *streaming* áudio por pedido (sem necessidade de interação em tempo real), a fiabilidade da rede fim-a-fim tem que atingir valores mínimos para que a experiência do utilizador seja agradável, ainda que o nível de exigência seja muito menor do que para sistemas de *streaming* com interação em tempo real.

O aumento contínuo na largura de banda disponível nos serviços de acesso à Internet e o aumento muito significativo do número de utilizadores dos serviços de *streaming* áudio acarreta desafios importantes. Ao contrário do *streaming* de conteúdo multimédia usado nos sistemas de *broadcasting* de canais televisivos ou canais de rádio, no *streaming on-demand* de coleções musicais (privadas ou não) não é útil a utilização de estratégias/protocolos

de rede/transporte de dados (como o *multicast*) que ajudem na poupança da largura de banda necessária para satisfazer um grupo elevado de utilizadores em simultâneo. Também a largura de banda fim-a-fim e a fiabilidade do serviço de acesso à Internet pode variar muito de utilizador para utilizador ou até para o mesmo utilizador, se, por exemplo, o ponto de acesso à Internet for móvel. Estes problemas costumam ser minimizados adotando as seguintes estratégias (individualmente ou em simultâneo):

LIMITAÇÃO DA QUALIDADE DO CONTEÚDO TRANSFERIDO - a maior parte dos serviços de *streaming* áudio que permitem a reprodução a pedido (ou por recomendação) de músicas duma coleção (privada ou não) limitam a qualidade do áudio transferido e disponibilizado na aplicação cliente do utilizador, independentemente da qualidade com que está disponível/armazenada no servidor (privado ou não); nos sistemas mais avançados, a qualidade pode variar dependendo da largura de banda disponível fim-a-fim, ainda que a qualidade máxima seja sempre limitada; esta limitação, dinâmica ou não, na qualidade do áudio transferido e reproduzido é uma das principais limitações destes sistemas;

IMPLEMENTAÇÃO DE *BUFFERS* NA REPRODUÇÃO ÁUDIO - neste caso, os *buffers* da aplicação cliente onde o áudio é reproduzido conseguem absorver parte dos problemas da flutuação do ritmo de dados disponível para o mesmo utilizador; a implementação destes *buffers* acarreta algum atraso no início da reprodução dum trecho musical, mas normalmente não degrada a experiência do utilizador com relevância;

IMPLEMENTAÇÃO DE SERVIDORES DE CACHE DO CONTEÚDO ÁUDIO - alguns sistemas de *streaming* áudio por pedido incluem servidores de cache intermediários entre os servidores principais onde as coleções originalmente estão armazenadas como forma de diminuir o atraso e tirar partido duma maior largura de banda disponível entre o servidor de cache e o utilizador do que entre o servidor principal e o utilizador; estes servidores de cache podem ser organizados numa arquitetura em árvore com a implementação de vários níveis de cache; os principais problemas desta estratégia são o custo dos elementos adicionais na arquitetura, a largura de banda consumida pelo próprio sistema de cache e a eventual utilização de conteúdo em cache com uma qualidade que pode ser inferior ao possível em determinada altura (nalguns casos este problema é resolvido mantendo em cache sempre cópias do áudio na maior qualidade possível no sistema, ainda que o custo de armazenamento e o gasto de largura de banda pelo próprio sistema de cache sejam superiores).

1.2 GESTÃO DAS COLEÇÕES

Um dos desafios destes serviços distribuídos é a criação e manutenção da meta-informação das coleções musicais que permita uma procura e identificação eficiente das músicas, dos artistas, dos álbuns, etc. Estes sistemas costumam usar duas estratégias distintas para organizar as coleções:

DEFINIR UMA ESTRUTURA PURAMENTE ORGANIZATIVA PARA A COLEÇÃO - nesta estratégia as coleções musicais são organizadas em árvores de nomeação em que cada tipo de informação é um nível; os topos/raízes das árvores costumam representar os tipos de música, enquanto o último nível (ou folhas) representam as músicas da coleção; apenas a este último nível se associam ficheiros áudio com as músicas; depois do primeiro nível dedicado ao tipo de música costuma aparecer um nível dedicado à nomeação/identificação do artista, banda, compositor, etc.; depois deste é comum haver um ou mais níveis dedicados à identificação do álbum, obra, coletânea, etc.; em seguida pode existir um nível dedicado nomeação de partes distintas dum álbum ou duma obra (por exemplo, quando um álbum é editado em vários CDs ou quando uma obra é dividida em vários volumes, atos, etc.); nesta estratégia é possível definir vários campos de informação associados ao tipo de informação em cada nível (para além dum nome) e os valores desses campos costumam ser diretamente dedutíveis dos mecanismos de armazenamento locais (sistema de ficheiros, por exemplo) ou de meta-informação embebida nos próprios ficheiros de áudio ou de ficheiros especiais de meta-informação armazenada em cada nó; além disso, esta estratégia é mais facilmente implementável sem recurso a software de bases de dados utilizando, por exemplo, a própria organização dos sistemas de ficheiros dos sistemas operativos onde as coleções estão armazenadas; isto permite, inclusive, que uma coleção privada possa ser mantida manualmente pelo utilizador; esta solução é a mais utilizada em sistemas quando a coleção privada do utilizador também é armazenada localmente e a sua gestão é manual ou através duma aplicação local de gestão de coleções musicais; uma limitação deste tipo de solução é a dificuldade de associar a mesma música (ou grupo de músicas, álbum, etc.) a mais do que um nível de informação sem duplicar os ficheiros áudio; nos serviços de *streaming* áudio remoto, em que as coleções (privadas ou públicas) são armazenadas e geridas pelo próprio serviço, esta alternativa não costuma ser viável e não é escalável;

DEFINIR UMA ESTRUTURA PURAMENTE ORGANIZATIVA PARA A COLEÇÃO - nesta estratégia as coleções musicais são puramente associativas, i.e., as músicas são identificadas por tabelas virtuais de associação entre ficheiros áudio armazenados algures no sistema (normalmente identificados por uma *tag*) e campos de qualificação de vários tipos (nome, tipo, artista, etc.); nesta solução, mais escalável, é obrigatório o uso de sistemas de bases de dados para gerir a meta-informação e as tabelas de associação; isto também facilita a

não duplicação de ficheiros áudio que estejam associados a mais do que um conjunto de valores dos campos de meta-informação; esta solução é a escolhida nos serviços de *streaming* áudio remoto e, geralmente, quando o utilizador regista a sua coleção privada no serviço remoto só é autorizado a registar as músicas da sua coleção que sejam identificáveis como músicas já pertencentes à coleção pública do próprio serviço; isto prende-se com a tentativa de manter a base de dados das coleções musicais de todos os utilizadores escalável e performante e também para evitar problemas eventuais de *copyright*.

De notar que a gestão simultânea das coleções de todos os utilizadores registados numa plataforma de *streaming* áudio é uma complexidade adicional que não está presente nos sistemas de *streaming* que são apenas usados localmente na rede do utilizador.

1.3 DIREITOS DE AUTOR

Com o desenvolvimento da Internet e com a crescente facilidade na partilha de conteúdos multimédia, em particular de ficheiros áudios, a indústria musical sofreu um impacto relevante. Os utilizadores podiam, sem grandes restrições, construir uma coleção musical recorrendo frequentemente a mecanismos ou serviços de legalidade duvidosa, como o serviço original Napster, com muitas práticas a infringirem direitos de autor ou de *copyright* e podendo ser mesmo classificadas de pirataria musical. Este período conturbado teve o seu ponto alto aquando da condenação judicial do Napster. A solução para este problema acabaria por surgir com serviços, como o iTunes ou o Spotify, que passaram a oferecer *streaming* musical gratuitamente em troca de consumo de publicidade ou a um preço muito acessível. Além disso, através destes serviços também passou a ser possível comprar e possuir legalmente uma cópia das músicas a um preço competitivo, quando comparado com a compra em lojas físicas tradicionais. A facilidade de acesso a estes serviços gratuitos ou de baixo custo e a possibilidade de poder aceder ao *streaming* de música em qualquer lugar com acesso à Internet garantiu o seu sucesso e diminuiu drasticamente a partilha não legal de conteúdo protegido.

Um problema menos discutido é a responsabilidade das plataformas que aceitam albergar coleções privadas dos seus utilizadores em proteger os direitos de autor das músicas que estão presentes nas coleções privadas dos utilizadores, mas não estão presentes na coleção de acesso público que essas plataformas têm autorização para partilhar. Para além de legalmente não poderem partilhar essas músicas têm também que garantir que a plataforma não é usada como meio para a partilha ilegal dessas músicas.

1.4 MOTIVAÇÃO

As plataformas de *streaming* permitiram que os artistas continuassem a criar e a lucrar com as suas obras através dos royalities cobrados por cada venda ou por cada música disponibilizada para *streaming*. Por sua vez, os utilizadores financiam as plataformas através de uma taxa de utilização, da compra definitiva de músicas e/ou pelo consumo de publicidade. Excetuando o caso da compra definitiva, os utilizadores não podem descarregar nem possuir qualquer cópia das músicas só podendo acede-las através do serviço de *streaming*. Em geral, no caso da compra definitiva o utilizador pode descarregar uma cópia das músicas adquiridas para a sua coleção privada armazenada localmente ainda que não esteja autorizado a partilha-las. Mesmo deixando de ser utilizador da plataforma, poderá continuar a possuir legitimamente as cópias das músicas, mas não poderá ouvi-las através do serviço de *streaming* remoto de que deixou de ser utilizador.

No caso dos utilizadores que possuam legitimamente coleções privadas de tamanho relevante e que ouçam sobretudo música das suas coleções, existem inúmeras soluções eficientes para o *streaming* na sua rede local. Estas soluções tanto são baseadas em tecnologias proprietárias de uso exclusivo dum fabricante (ou dum grupo de fabricantes) de equipamentos multimédia como são baseadas em tecnologias normalizadas como, por exemplo, o Digital Living Network Alliance (DLNA) [3] ou o Universal Plug aNd Play (UPnP) [4]. O problema destas soluções é que apenas estão disponíveis entre dispositivos da mesma rede local não sendo utilizadas para *streaming* de áudio remoto.

Como já foi referido, para os utilizadores que possuam coleções privadas de tamanho relevante (dezenas de milhar de músicas ou mais) o acesso à sua coleção em qualquer lugar através dum dispositivo com acesso à Internet pode, ainda hoje, ser um desafio complicado de resolver. A solução de ter cópias da sua coleção em todos os dispositivos ou ter uma cópia num único dispositivo de armazenamento que se conecta ao dispositivo computacional que estiver a ser usado não é prático ou pouco viável, dependendo do dispositivo utilizado.

A solução mais prática e segura será sempre a utilização dum serviço que aceda remotamente à sua coleção através de *streaming* áudio. Para os utilizadores poderem usar as plataformas atuais de *streaming* terão de transferir uma cópia real ou uma cópia virtual da sua coleção para os servidores remotos dessas plataformas. Estas transferências são pouco práticas no caso de coleções grandes e, na maior parte dos casos, o acesso à coleção é feito através duma qualidade máxima que é definida pela plataforma e não pela qualidade dos ficheiros áudio originais da coleção nem pela largura de banda disponível no ponto de acesso ao serviço. Adicionalmente, estas plataformas obrigam à utilização de aplicações de reprodução musical próprias o que pode ser limitativo nalguns tipos de dispositivos e de sistemas operativos. Além disso, na maior parte dos casos, as plataformas não garantem o registo correto de todas as músicas incluídas na coleção original, dependendo do sistema

de nomeação e gestão de registos. Normalmente estes sistemas só garantem o registo correto de músicas armazenadas num determinado formato contendo algumas *tags* de meta-informação mínima ou de músicas adquiridas através da própria plataforma. Também é normal estes serviços de *streaming* não permitirem a recuperação da coleção original exata através da transferência em sentido contrário, isto é, dos servidores da plataforma para o armazenamento de qualquer dispositivo local do utilizador.

Uma solução que os utilizadores de grandes coleções privadas por vezes adotam é o armazenamento dos ficheiros áudio das suas coleções em sistemas de armazenamento *cloud*. Nesta solução, no entanto, a gestão dos ficheiros áudio não é prática e o acesso às coleções faz-se usando sistemas de armazenamento virtual que pode não ser compatível com a ferramenta de reprodução áudio utilizada (que pode variar dependendo do dispositivo que o utilizador esteja a usar em determinado momento). Além disso, esta solução pode ser problemática em termos de segurança quando o utilizador quiser aceder à sua coleção em dispositivos de terceiros. Por fim, salienta-se que, ainda que alguns serviços de armazenamento na *cloud* já ofereçam serviços de *streaming* quando reconhecem os ficheiros como sendo áudio, a maior parte só permite que os ficheiros sejam armazenados e acedidos como ficheiros, sem qualquer semântica aplicacional. Ou seja, os ficheiros áudio são acedidos sem se utilizarem técnicas específicas para reprodução áudio remota, como por exemplo, a adaptação da qualidade da reprodução mediante a qualidade de serviço da rede de acesso à Internet.

Atendendo aos tipos de serviços de acesso remoto a coleções de música mais comuns atualmente, surgiu a ideia de tentar definir um serviço que permitisse o *streaming* áudio remoto de coleções de utilizadores em redes privadas, sem que os utilizadores tenham de transferir a sua coleção para servidores remotos de plataformas de *streaming* áudio ou de sistemas de ficheiros *cloud*. Esta solução deveria ser passível de implementação utilizando apenas tecnologias de utilização aberta e livre e protocolos normalizados.

1.5 OBJETIVOS

Tendo em consideração a introdução apresentada nas secções anteriores, definiu-se o seguinte conjunto de objetivos principais no início dos trabalhos da dissertação, dividindo o trabalho em três fases distintas:

- Estudo dos vários tipos de serviços prestados pelas plataformas atuais de *streaming* áudio; análise de tecnologias normalizadas utilizadas no *streaming* áudio; identificação e análise de trabalhos de investigação e desenvolvimento na área de *streaming* áudio; identificação das principais funcionalidades, diferenças relativas e limitações dos sistemas para utilizadores com coleções musicais com elevado número de ficheiros áudio; análise de tecnologias normalizadas utilizadas no *streaming* áudio; breve dis-

cussão sobre trabalhos de investigação e desenvolvimento na área de *streaming* áudio, nomeadamente os desenvolvidos na Universidade do Minho;

- Definição duma arquitetura para uma plataforma de *streaming* áudio direcionada para utilizadores que pretendem acesso remoto à sua coleção privada em qualquer tipo de dispositivo fixo ou móvel com acesso à Internet; a arquitetura deveria permitir utilizar tecnologias e protocolos normalizados, atuais e de livre acesso; a especificação funcional deste serviço deveria permitir o *streaming* remoto sem necessidade de transferir a coleção privada do utilizador para servidores externos de terceiros, fora da sua rede privada; estes requisitos devem fazer com o que o sistema seja prático de gerir e usar mesmo para utilizadores que já tenham grandes coleções musicais em formato digital; para além disso, o modelo deve permitir construir uma aplicação para reprodução musical para *streaming* áudio independente do tipo de dispositivo, do sistema operativo e do tipo de servidores de *streaming* e de armazenamento; mais ainda, a arquitetura deve proteger as coleções dos seus utilizadores pelo que não devem ser transferidas cópias definitivas das músicas para nenhum servidor ou dispositivo fora da rede privada do utilizador, ainda que possam ser implementados mecanismos de cache; os requisitos funcionais serão depois discutidos com mais detalhe no capítulo 3;

- Desenvolvimento e teste dum sistema protótipo que implemente os principais requisitos funcionais definidos para arquitetura teórica; esta plataforma protótipo deve incluir um sistema local para armazenamento e gestão da coleção privada do utilizador, um sistema remoto que implemente um serviço integrado da gestão de utilizadores e de *streaming* áudio direto e indireto entre o servidor privado do utilizador e o dispositivo de acesso remoto na Internet, e uma aplicação reprodutora de coleções musicais para poder ser utilizada em dispositivos móveis com acesso à Internet; os testes realizados ao protótipo deveriam permitir estabelecer a viabilidade e validade da arquitetura em atingir os objetivos propostos, para além de permitir analisar com mais propriedade as eventuais limitações da arquitetura teórica ou da própria plataforma protótipo desenvolvida.

1.6 Organização do documento

Depois desta introdução, o capítulo 2 discute algumas tecnologias que permitem implementar os vários tipos de solução para *streaming* áudio, servindo de base a muitas das aplicações mais utilizadas hoje-em-dia para consumo de conteúdo musical. No capítulo 3 será apresentada a arquitetura da solução proposta para *streaming* áudio remoto a pedido de coleções armazenadas localmente na rede privada dos utilizadores. No capítulo seguinte serão descritas as estratégias, as tecnologias e os protocolos adotados para produzir um

sistema protótipo implementando as principais funcionalidades da arquitetura definida no capítulo anterior e discute-se a validade e viabilidade da solução proposta. Por fim, o capítulo 5 é dedicado às conclusões da dissertação, pondo em evidência quais os objetivos alcançados e as limitações identificadas, sugerindo-se melhorias para uma nova versão do protótipo.

2

TECNOLOGIA E TRABALHOS RELACIONADOS

Os primeiros esforços de implementação de sistemas de *streaming* áudio foram realizados no contexto dos sistemas de distribuição de áudio (e posteriormente de vídeo) entre dispositivos duma rede local. Estes sistemas, muitas vezes integrados em redes domóticas, eram dedicados à distribuição de fontes de informação multimédia externas dentro da rede local (distribuição de rádio digital, serviços de *streaming* áudio e até de distribuição de canais televisivos). No campo do *streaming* áudio apenas, para além das soluções totalmente proprietárias, como a solução da Sonos – uma das mais antigas e mais conhecidas, o esforço comunitário mais independente apareceu através do fórum UPnP com o seu próprio conjunto de protocolos e mecanismos básicos. Pouco depois começaram a aparecer soluções mais completas e complexas, impulsionadas por grandes empresas como a Sony ou a Apple, baseadas em protocolos de uso mais aberto como o DNLA ou o DAAP, ainda que tendo como base a família de protocolos UPnP.

Em qualquer dos casos, estas tecnologias permitiram o desenvolvimento de soluções para *streaming* áudio em redes domésticas/locais, mas não eram dedicadas para *streaming* áudio através de dispositivos remotos com acesso à Internet. Nesta vertente, as primeiras plataformas impactantes a nível global foram o serviço Napster, primeiro, e depois o serviço iTunes, ainda que as respetivas tecnologias de *streaming* áudio propriamente ditas fossem rudimentares, ou seja, eram mais um serviço de venda, partilha e distribuição de ficheiros áudio digitais do que um verdadeiro serviço de *streaming*.

Com as melhorias na largura de banda disponível na Internet foram surgindo inúmeras verdadeiras plataformas de *streaming* áudio remoto (outras, como a iTunes, foram-se modernizando até gerarem mais uma aplicação adicional e específica para *streaming* áudio, o Apple Music). A plataforma que rapidamente conseguiu uma posição de destaque no mercado foi o Spotify. Inicialmente este tipo de plataformas não permitia o armazenamento nos seus servidores das coleções privadas dos seus utilizadores, mas, recentemente, cada vez mais plataformas incluem esta funcionalidade, ainda que com algumas limitações, como já foi referido no capítulo anterior.

Em qualquer dos casos, nenhuma das soluções anteriores permite a gestão e *streaming* áudio remoto de coleções privadas armazenadas nos dispositivos locais da rede doméstica dos utilizadores.

Tendo em consideração estes tipos de sistema de distribuição de áudio, neste capítulo serão apresentadas e comparadas tecnologias que permitem efetuar o *streaming* de áudio através duma rede local e entre vários dispositivos, para além das principais plataformas de *streaming* musical remoto, a pedido, que estão atualmente em voga na Internet.

Por fim, será mencionado um trabalho de investigação e desenvolvimento relacionado. É um trabalho de cariz académico e foi desenvolvido na Universidade do Minho, ainda que a arquitetura definida possa servir de base a futuras soluções profissionais (comerciais ou não, mas sempre com suporte de tecnologias abertas e gratuitas).

2.1 UPnP, DLNA e DAAP

2.1.1 *UPnP*

A tecnologia UPnP foi promovida pela UPnP Forum [5] e é muito mais do que uma extensão do modelo *plug and play*, na medida em que, foi concebido para não existir qualquer configuração do aparelho e promover uma descoberta automática de outros. Quando se ligam a uma rede local, os dispositivos estabelecem uma conexão automatica entre si, partilhando as suas capacidades funcionais.

UPnP v1 vs UPnP v2

Inicialmente o UPnP foi lançado pela UPnP Forum dando origem à versão 1. Recentemente, em 2016, todos os esforços relacionados com o UPnP passaram a ser controlados pela Open Connectivity Foundation (OCF), dando origem à versão 2.

Não existem quaisquer diferenças em termos da arquitetura dos dispositivos genéricos e dos protocolos de configuração apresentados na arquitetura da versão 1[5] e na arquitetura da versão 2[6]. No entanto, há uma evolução na definição dos dispositivos de AV. no que toca à interoperabilidade entre os dispositivos e no controlo de acesso, proteção de conteúdo e nos direitos de autor. No entanto, a documentação oficial não é clara quanto às diferenças na implementação das diferentes funcionalidades e nos objetivos concretos que são procurados pelas alterações ao utilizar uma tecnologia como o DLNA, já vem implementado um protocolo que tem como objetivo colmatar este aspeto.

Sendo uma versão mais recente, a versão 2 do UPnP [7] é capaz de suportar mais formatos nos diferentes clientes em relação à versão 1[8], já que também é uma versão mais recente.

Por exemplo, o UPnP:1 [9] utiliza o *daemon*, *linux-igd*, ao passo que o UPnP:2 usa *miniupnd*[10], que lhe permite dar suporte a NATPMP (é um protocolo de rede para estabelecer configurações de conversão de endereço de rede (NAT) e configurações de encaminhamento de porta automaticamente sem esforço do utilizador [11]).

Devido às diferenças apresentadas, e considerando o contexto desta dissertação, as mesmas não são relevantes, pelo que o resto da informação apresentada baseia-se na documentação existente para a versão 1 do UPnP.

UPnP Device Architecture

O processo de configuração contém vários passos [5]: *addressing*, *discovering*, *descriptions*, *controls*, *events* e *presentation*. São estas configurações que permitem aos dispositivos, numa mesma rede local, contactar entre si. As suas descrições serão apresentadas de seguida:

0. *Addressing* – é o passo zero do *UPnP Networking*. Cada dispositivo deve de ter um cliente *Dynamic Host Configuration Protocol* (*DHCP*) e ser capaz de procurar por um servidor DHCP aquando da primeira conexão à rede. Nos casos em que o próprio dispositivo já possui um servidor DHCP então deve escolher um endereço dos valores que o mesmo alberga. Retomando o caso inicial, se estiver disponível um servidor DHCP então o dispositivo deve usar o endereço que lhe foi atribuído. Caso contrário estamos perante um cenário em que a rede não é controlada e, portanto, o dispositivo deve iniciar um processo denominado de *Auto-IP* em que se autoatribui um endereço IP.

1. *Discovering* – assim que um dispositivo possui um endereço, o próximo passo, numa rede UPnP, é o descobrimento de novos serviços. O protocolo de descoberta é conhecido como *Simple Service Discover Protocol* (*SSDP*). Este protocolo permite que, um dispositivo conectado à rede seja capaz de anunciar os seus serviços. Para tal envia mensagens de *multicast* que são recebidas pelos pontos de controlo de forma a que estes possam atualizar a sua lista de serviços. O mesmo processo acontece quando se adicionam pontos de controlo à rede. Todos os dispositivos que receberam a mensagem devem responder, caso as suas caraterísticas coincidam com os critérios de procura. O objetivo destas mensagens é a partilha da informação com algumas caraterísticas de cada dispositivo ou dos seus serviços, como por exemplo, tipo, identificador e uma localização (apontador, URL). A figura 1 representa o processo referido acima.

2. *Descriptions* – quando um ponto de controlo descobre um dispositivo a informação que tem do mesmo é escassa (apenas tem os dados fornecidos na mensagem de descoberta, que foram apresentados anteriormente). Desta forma, o ponto de controlo tem de extrair a informação da descrição do dispositivo fornecida no *URL* de acesso. Esta é divida em duas partes: descrição do dispositivo, onde se encontra informação a cerca do fornecedor (nome, número de série, outros URLs com mais informação, etc) e descrição dos serviços, onde são descritas as capacidades do dispositivo. Cada elemento de informação é constituído pelo nome e tipo de serviço e ainda 3 URLs, um para a descrição do serviço, outro para o controlo e o último para eventos (estas últimas

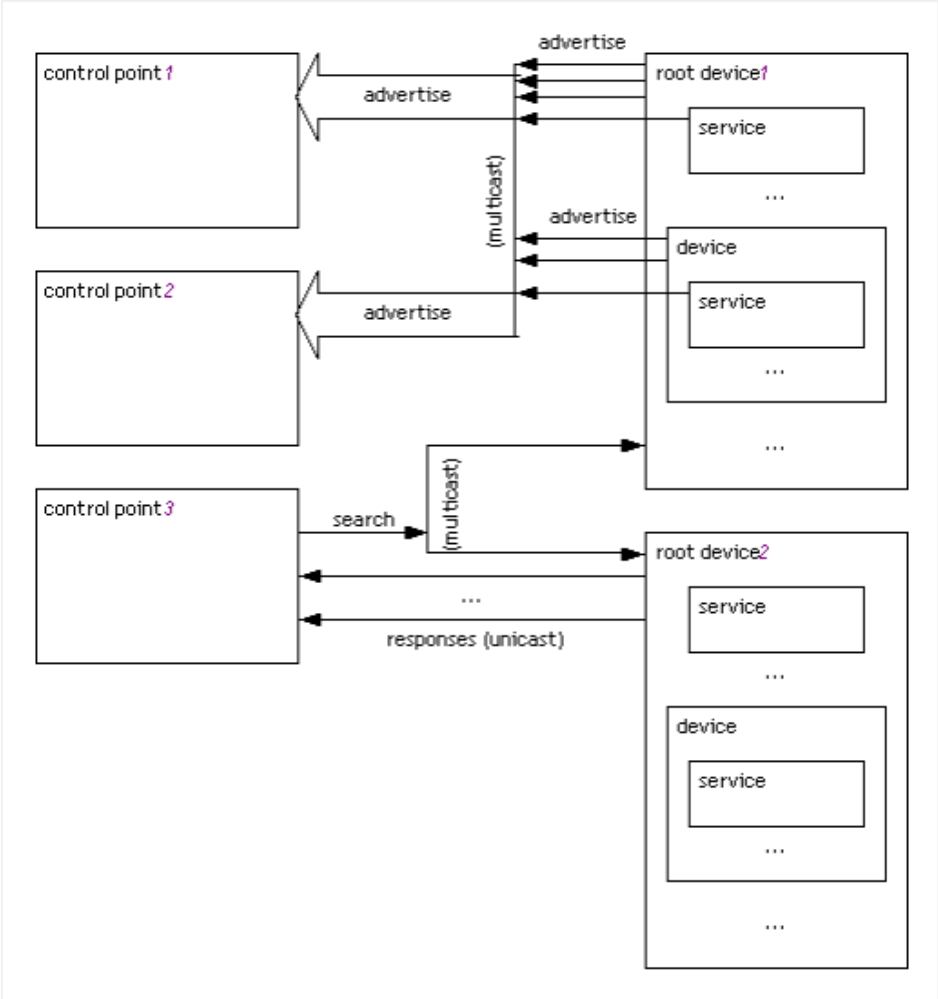


Figura 1: Fase Discovering - UPnP [9]

duas propriedades serão detalhadas mais adiante). De salientar que na descrição do serviço está presente uma lista de ações a que o serviço pode responder, que o serviço responde, e argumentos (variáveis) para cada ação. Para obter a descrição do dispositivo o ponto de controlo envia um pedido HTTP GET para o URL na mensagem de *discovery* e espera pela resposta. A figura 2 apresenta um esquema do mesmo.

3. *Controls* – uma vez obtida a descrição de um dispositivo, um ponto de controlo pode realizar uma ação sobre o dispositivo. Para tal tem de enviar uma mensagem para o URL de controlo transmitido na mensagem de *description*. A resposta do serviço pode conter o resultado obtido ao executar a ação ou então qual o erro ocorrido. Invocar ações é como executar uma *remote procedure call* (chamada remota de procedimento, em tradução literal). São mensagens XML e usam o Simple Object Access Protocol (SOAP). De salientar que, se os dados forem demasiado grandes não devem constar

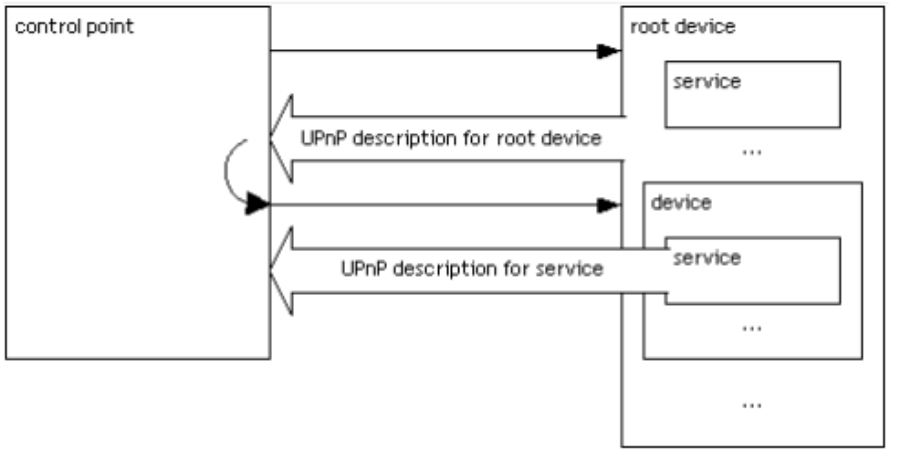


Figura 2: Fase Descriptions - UPnP [5]

diretamente na mensagem SOAP, mas sim num URL para depois se realizar um pedido HTTP para extração dos mesmos.

4. *Events* – após os passos 1 (*discovering*) e 2 (*description*) o ponto de controlo tem todas as condições reunidas para iniciar eventos (*events*). O protocolo de notificação por eventos, utilizado no UPnP é o General Event Notification Architecture (GENA). O serviço publica atualizações através de mensagens de eventos quando há uma mudança das variáveis. O ponto de controlo recebe estas mensagens, caso tenha subscrito. Para tal, tem de enviar uma mensagem de subscrição para o dispositivo desejado. Numa primeira mensagem de evento para o ponto de controlo, vão todas as variáveis e os seus respetivos valores para que o mesmo possa obter toda a informação. As restantes mensagens, para o mesmo ponto de controlo, contêm apenas os dados das variáveis modificadas.

5. *Presentation* – por fim, apresentação (*presentation*), é o último passo processo. Se na mensagem *description* estiver presente um URL para a *presentation*, então é possível ao ponto de controlo aceder ao mesmo, carregar a página para um navegador de Internet e, dependendo das funcionalidades da página, permite ao utilizador controlar ou observar o estado do dispositivo.

UPnP AV Architecture

A Arquitetura AV UPnP [9] define a interação existente entre os pontos de controlo UPnP e os dispositivos AV UPnP. É independente de qualquer tipo de dispositivo, formato ou protocolo de transferência. Num cenário UPnP que não envolva a transferência de conteúdo, o ponto de controlo é responsável por controlar cada dispositivo. Apesar de o ponto de

controlo ser capaz de administrar vários dispositivos, a interação encontra-se isolada entre as duas entidades. Isto é, para cada dispositivo, o ponto de controlo cria uma conexão e nessa apenas comunicam os dois componentes presentes.

Num cenário AV (termo usado na documentação para se referir a um cenário com transferência de conteúdo multimédia) existe um fluxo de conteúdo que deve ser transportado de um dispositivo para outro, obrigando a que os intervenientes comuniquem diretamente entre si, utilizando para tal um protocolo à parte do UPnP. Porém, o ponto de controlo tem um papel importante antes de se iniciar a transferência, já que o mesmo é responsável por inicializar e configurar os dois dispositivos para que o conteúdo possa ser transferido entre ambos, i.e., ele é responsável por iniciar a transferência, mas não tem qualquer papel durante o decorrer deste processo.

Na imagem 3 está presente um esquema sobre a interação entre as diferentes entidades num cenário AV. A Arquitetura AV UPnP suporta diferentes tipos de dispositivos (TVs, PCs, câmeras, etc) e múltiplos formatos (JPEG, MP3, WMA, etc).

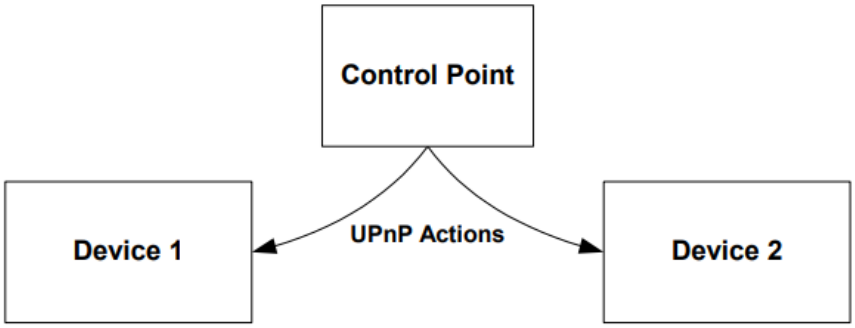


Figura 3: Modelo de Interação dos Dispositivos UPnP [9]

Existem três entidades diferentes que estão envolvidas num cenário AV: o Ponto de Controlo, MediaServer (a fonte do conteúdo) e o MediaRenderer (cliente final). Esta é a configuração mais comum, mas, no entanto, a Arquitetura AV UPnP é flexível, no sentido em que, permite que um mesmo dispositivo não se restrinja a uma única função. Por exemplo, um dispositivo MP3 pode servir como *MediaRenderer* que vai reproduzir as músicas transferidas para o mesmo, como também pode servir de Ponto de Controlo permitindo controlar o conteúdo (pausar, avançar ou recuar nas músicas, por exemplo).

O principal papel de um utilizador nestes cenários é poder reproduzir conteúdo num dispositivo específico (desde que tenha a capacidade de o reproduzir, i.e., seja um *MediaRenderer*). O utilizador deve interagir com a interface do Ponto de Controlo (isto se esta existir, ficando à consideração dos produtores do dispositivo defini-la) para localizar e selecionar o conteúdo desejado de um *MediaServer* e escolher, de seguida, onde quer que o seu conteúdo seja exibido, ou seja, selecionar qual o dispositivo *MediaRenderer* desejado.

De salientar que as comunicações são livres. Nestes casos, se seguirem o protocolo UPnP, os dispositivos trocam mensagens SOAP através de HTTP. Porém é dada a liberdade aos produtores dos dispositivos para, caso assim o entendam, criar um protocolo de comunicação privado. Mas com a ressalva de que estes dispositivos apenas conseguem comunicar com outros que entendam este protocolo. Mais uma vez, durante a transferência de conteúdo entre o *MediaServer* e o *MediaRenderer* não há qualquer limitação quanto ao protocolo a ser tido em conta, ficando à consideração dos dispositivos escolher o que melhor se adequa a ambos.

MediaServer

Um dispositivo MediaServer contém ou tem acesso a uma variedade de conteúdos multimédia, quer estejam guardados localmente ou acessíveis a partir de um outro dispositivo. Tem a capacidade para aceder ao seu conteúdo e transmiti-lo para outro dispositivo. Assim que solicitado, dá permissões para que o Ponto de Controlo procure os itens que disponibiliza para o utilizador reproduzir. Um MediaServer poderá ter incorporado até três serviços distintos:

CONTENT DIRECTORY SERVICE – este serviço permite ao Ponto de Controlo identificar o conteúdo existente no dispositivo. A principal ação é *Broswe()* devolvendo informação detalhada sobre cada item do conteúdo disponibilizado. Permite ainda obter informação sobre que protocolos de transferência e formatos de dados são suportados.

CONNECTION MANAGER SERVICE – este serviço serve para controlar as conexões de um determinado dispositivo. A principal ação é *PrepareForConnection()*. Retorna como resultado um identificador único da conexão, *InstanceID*. Pode ainda, dependendo da implementação, retornar um outro identificador, *AVTransport InstanceID*, que permite ao Ponto de Controlo gerir o fluxo do conteúdo (parar, pausar, etc). De notar que, caso o dispositivo não seja capaz de efetuar múltiplas transferências, então não necessita de implementar esta ação, dado que, por defeito, como só possuirá uma conexão, à mesma será atribuído um identificador único com o valor zero. Outra ação disponível é *ConnectionComplete()* que deve ser invocada quando o Ponto de Controlo termina uma conexão com este dispositivo.

AV TRANSPORT SERVICE – este serviço é usado para que o Ponto de Controlo possa ter controlo sobre o comportamento da reprodução de conteúdo associado com um específico *AVTransport*. Dependendo, mais uma vez, do dispositivo, este serviço pode ou não ser suportado.

Media Render

Um dispositivo MediaRender obtém o conteúdo de um dispositivo MediaServer através da rede. A sua principal caraterística é permitir ao Ponto de Controlo gerir como o conteúdo é reproduzido (através do controlo de parâmetros como o brilho, volume, etc). O tipo de conteúdo que um MediaRenderer recebe depende dos protocolos de transporte e dos formatos dos dados que suporta. Pode ainda permitir que um utilizador consiga controlar o fluxo dos conteúdos (parar, pausa, etc). Um MediaRenderer, à semelhança de um dispositivo MediaServer, tem até três serviços distintos disponíveis:

RENDERING CONTROL SERVICE – permite que o Ponto de Controlo manipule a reprodução de um certo conteúdo, controlando vários parâmetros, como por exemplo, o brilho, contraste, volume, etc.

CONNECTION MANAGER SERVICE – à semelhança do do que é possível no MediaServer, permite controlar as conexões associadas a este serviço. Possui igualmente as ações, *PrepareForConnection()* e *ConnectionComplete()*, construídas com o mesmo prepósito das respetivas ações no MediaServer. Para além destas, possui uma outra ação, *GetProtocolInfo()*, que permite que um Ponto de Controlo obtenha os protocolos de transferência e os formatos de conteúdos suportados pelo dispositivo.

AV TRANSPORT SERVICE – permite que um Ponto de Controlo manipule a reprodução de um conteúdo, sendo possível parar, reproduzir, procurar, etc. Depende da implementação do dispositivo se este serviço está ou não está disponível.

Ponto de Controlo

O Ponto de Controlo coordena a operação entre o MediaServer e o MediaRenderer. Normalmente isto é conseguido por comando de um utilizador que interage com este dispositivo através de uma interface disponibilizada pelo mesmo. Um Ponto de Controlo tem de implementar uma série de funcionalidades para interagir com os dispositivos:

1. Discover AV Devices – utiliza o processo UPnP's Discovery, para identificar os dispositivos MediaServer e MediaRenderer na rede local;
2. Located Desired Content – utiliza as ações disponibilizadas pelo MediaServer, por exemplo a *Browse()*, para localizar o conteúdo;
3. Get Renderer's Supported Protocols/Formats – utiliza o serviço *Connection Manager*, através da ação *GetProtocolInfo()*, disponibilizado num dispositivo MediaRender, para obter informação sobre os formatos de dados e os protocolos suportados;
4. Compare/Match Protocols/Formats – compara as informações obtidas através das funcionalidades 2 e 3, encontrando os possíveis pares de dispositivos compatíveis;

5. Configure Server/Renderer – utiliza a ação, *PrepareForConnection()*, disponibilizada no serviço *Connection Manager* em ambos os dispositivos para poder controlar o fluxo de reprodução dos conteúdos. De notar que como esta ação é opcional nos dispositivos, caso não seja fornecido qualquer valor, então o Ponto de Controlo tem de usar o valor por defeito para o identificador, que, neste caso, é zero;

6. Select Desired Content – utiliza o serviço *AVTransport* (se foi obtido o identificador através da funcionalidade 5) e invoca a ação *SetAVTransportURI()* para identificar cada elemento do conteúdo a ser transferido;

7. Start Content Transfer – utiliza o serviço *AVTransport* para que o utilizador possa controlar a reprodução, invocando a ação correspondente ao pedido de pausa, parar, etc;

8. Adjust Rendering Charateristics - utiliza o serviço *Rendering Control* de um dispositivo MediaRender para que o utilizador possa controlar as caraterísticas da reprodução, invocando a ação correspondente ao controlo de brilho, volume, etc;

9. Repeat: Select Next Content – utiliza tanto as ações *SetAVTransportURI()* e *SetNextAVTransportURI()*, disponíveis no serviço *AVTransport*, para identificar o próximo elemento do conteúdo a ser transferido de um MediaServer para um MediaRenderer, que se encontram conectados. Repete este procedimento conforme for necessário.

10. Cleanup Server/Renderer – quando a sessão termina, para o contexto da sessão, os dispositivos envolvidos já não são necessários. Para tal, invoca-se a ação *ConnectionComplete()* disponível no serviço *ConnectionManager* de ambos.

Problemas de Segurança

Como o UPnP é um protocolo desenhado para utilizar em rede, a segurança é sempre um ponto crucial para os utilizadores aquando do uso das funcionalidades que o protocolo lhes dá. Porém, o UPnP não implementa qualquer tipo de autenticação, permitindo que entidades maliciosas possam aceder ao *router* e a partir daí obter acesso a todos os dispositivos presentes na rede local.

Convém relembrar que o UPnP foi desenhado para ser utilizado num ambiente doméstico. Desta forma, não existe qualquer tipo de função ou ação para verificar se o IP que está a ser usado realmente está inserido na rede doméstica. Outro problema é o facto de os dispositivos comprometidos na rede local podem albergar programas maliciosos que podem aceder a todos os outros dispositivos a partir do UPnP. Por isso estabelecer mecanismos adicionais de segurança é uma das prioridades no uso deste protocolo. Nestes casos é aconselhável que não se dê grandes privilégios (por exemplo, de administrador) aos Pontos de Controlo. Além disso, com o objetivo de reduzir o risco de ataques por injeção de mensagens através

de *cross-site scripting*. A UPnP Forum aconselha que dispositivos e Pontos de Controlo devam usar números aleatórios para as portas [12].

Devido aos cenários que existem que permitem a exploração das limitações da segurança do UPnP, os dispositivos devem implementar o *Device Protection Service* que suporta autenticação e controlo de acesso para dispostivos UPnP [13] e também *Device Security Service* que acrescenta uma autenticação forte, autorização, prevenção e privacidade nas ações SOAP do UPnP [14]. Existem outras soluções para aumentar a segurança do UPnP, porém não são *standard* pelo que fica a cargo dos utilizadores confiar ou não nos serviços prestados.

A publicação [15] alerta para o perigo associado à falta de segurança do UPnP. Neste caso, numa das maiores "guerras" de subscritores da história do *youtube*, uns utilizadores suecos, para além de enviarem imenso conteúdo para impressoras, conseguiram acederam a televisões e dispositivos de multimédia para reproduzir vídeos. Utilizaram as vulnerabilidades do UPnP para conseguir aceder a dispositivos *Chromecast* [16].

Para os utilizadores que não têm qualquer vantagem em utilizar as funcionalidades prestadas pelo UPnP então, por razões de segurança, devem desabilitar o uso deste protocolo quer no *router* quer nos dispositivos. Para os restantes, devem certificar-se que, pelo menos, os dispositivos que possuem têm as implementações de segurança standard para o UPnP referidas anteriormente e devem manter atualizados os dispositivos com o *firmware* mais recente.

2.1.2 *DLNA*

O *Digital Living Network Alliance* (*DLNA*) [17] tem como principal objetivo garantir a interoperabilidade entre dispositivos numa rede local, de modo a que seja possível a troca de conteúdos multimédia (imagem, som ou vídeo) através duma rede local. Para tal, foi idealizado um conjunto de diretrizes baseadas em padrões tecnológicos já existentes. Este padrão foi estabelecido pela SONY de modo a que outras empresas (i. e. empresas interessadas em utilizar esta tecnologia) produzissem os seus dispositivos de acordo com estas normas e pudessem assim garantir a interoperabilidade dos dispositivos na distribuição de conteúdos áudio. Os dispositivos têm de passar por uma série de testes para receber uma certificação que os permita identificar como um DLNA Device.

O *DLNA* divide os dispositivos multimédia em doze subclasses certificadas que estão distribuídas por três classes distintas, Home Network, Mobile Handled e Home Infrastructure, que serão descritas na secção Dispositivos DLNA. A classe de um dispositivo é determinada não pelo tipo de dispositivo em si, mas sim pelas capacidades e/ou caraterísticas que o mesmo apresenta. O DLNA é flexível pois um dispositivo pode estar presente em várias classes pelo facto de apresentar caraterísticas complementares. Todos os dispositivos utili-

zam o *Universal Plug and Play* (*UPnP*) como protocolo de descoberta e comunicação numa rede local.

Sucintamente, o funcionamento do DLNA pode ser descrito da seguinte maneira: o utilizador tem que ter instalado um sistema computacional com a certificação de *software* DLNA e que implementará um servidor multimédia. Para além disto, tem de possuir pelo menos um reprodutor de multimédia (por exemplo, uma televisão), também com uma certificação de *software* DLNA, que será capaz de obter o conteúdo existente no computador, apresentá-lo e reproduzi-lo (neste caso através de *streaming*). Pode ser adicionado ainda um controlador, como um *tablet* ou um *smartphone*, que obtém o conteúdo do computador e envia para a televisão, para que esta o reproduza. Com este controlador o utilizador tem um maior controlo sobre a reprodução de conteúdo. [18] [19]

Dispositivos DLNA

Cada dispositivo DLNA possui um conjunto de funcionalidades que vão definir o seu comportamento, capacidades e caraterísticas, o que é designado por um perfil do dispositivo (*device profile*).

O DLNA tem de correr em múltiplos aparelhos e perante tanta diversificidade é difícil que exista uma implementação *standard* que funcione corretamente em todos os dispositivos. Assim, no perfil do dispositivo podem ser adicionadas novas caraterísticas e capacidades, que vão definir um novo comportamento que pode ser diferente das funcionalidades base do DLNA.

No caso do *software* Flex [20], por exemplo, é permitido ao utilizador construir perfis DLNA, designando-se de *client profile*. No entanto, os dispositivos possuem um perfil de fábrica, o *system profile*, que deve de ser atualizado sempre pelo sistema do servidor de multimédia. Este não deve permitir ao utilizador comum alterar as especificações para que o mesmo não cause qualquer funcionamento incorreto.

Outro exemplo da flexibilidade disponível na definição dos dispositivos DLNA é o Commercial Video Profile (CVP), VidPath, que é uma extensão dos dispositivos DLNA e que permite a circulação de conteúdo publicitário. Nos países onde este tipo de conteúdo possa ter restrições este pode ser adptado. [21]

O DLNA divide os dispositivos em três diferentes classes: Home Network, Mobile Handled e Home Infrastructure Devices [22]. Cada classe inclui ainda diversas subclasses.

A classe Home Network Devices (HND) possui cinco subclasses:

- Digital Media Server (DMS) – estes dispositivos são responsáveis por armazenar e disponibilizar o conteúdo na rede. Certos dispositivos são ainda responsáveis pela proteção da informação. Exemplos: computadores e dispositivos NAS;

- Digital Media Player (DMP) – estes dispositivos procuram o conteúdo nos DMS e têm ainda a capacidade de renderização e de reprodução repetida. Exemplos: televisões, consolas, etc;
- Digital Media Render (DMR) – reproduzem o conteúdo (encontrado pelo DMS) recebido do DMC. Distingue-se do DMP pelo facto de não serem capazes de encontrar o conteúdo na rede, mas sim através de outra entidade, neste caso um DMC. Exemplos: televisões, recetores de áudio/vídeo, etc;
- Digital Media Controller (DMC) – procuram o conteúdo nos dispositivos DMS e reproduzem-nos nos DMR. Tratam das configurações das conexões entre os dispositivos DMS e os dispositivos DMR. Exemplos: *tablets*, câmeras digitais com *wi-fi*, etc;
- Digital Media Printer (DMP) – são dispositivos que acrescentam à rede o serviço de impressão. Exemplos: *networked all-in-one printers*, etc.

A classe Mobile Handled Devices (MHD) integra também 5 subclasses:

- Mobile Digital Media Server (M-DMS) – estes dispositivos sem fios são responsáveis por armazenar e disponibilizar o conteúdo na rede. Exemplos: *smartphones, music players* portáteis, etc;
- Mobile Digital Media Player (M-DMP) – encontram o conteúdo e reproduzem-no no DMS ou M-DMS. Exemplos: *smartphones*, *tablets*, etc;
- Mobile Digital Media Uploader (M-DMU) – estes dispositivos enviam (*upload)* o conteúdo para um DMS ou M-DMS. Exemplos: câmeras digitais e *smartphones*;
- Mobile Digital Media Downloader (M-DMD) – estes dispositivos sem fios encontram e transferem (*download*) o conteúdo exposto por um M-DMS ou DMS. Exemplos: *smartphones e music players* portáteis;
- Mobile Digital Media Controller (M-DMC) - procuram o conteúdo nos dispositivos DMS ou M-DMS e reproduzem-nos nos DMR. Tratam das configurações das conexões entre o *server* e o *render*. Exemplos: *smartphones* e *personal digital assistants* (PDA).

Por fim, a classe Home Infrastructure Devices (HID) possui apenas 2 subclasses:

- Mobile Network Connectivity Function (M-NCF) – são dispositivos responsáveis por criar um ponto de ligação (*bridge)* entre a conectividade dos MHD e a conectividade dos HND. Exemplos: *routers* e pontos de acesso;
- Media Interoperability Unit (MIU) – estes dispositivos providenciam a transformação do conteúdo dos formatos de multimédia requeridos entre os HND e MHD.

Arquitetura DLNA

O *DLNA* possui uma arquitetura que é dividida em diversas camadas. Na tabela 1 são apresentadas as mesmas, bem como os protocolos utilizados e as suas funcionalidades.

Tabela 1: Arquitetura DLNA

Camada	**Protocolo**	**Funcionalidades**
Link Protection	DTCP-IP	Como é que o conteúdo comercial é protegido na rede local
Media Format	JPEG,PNG/LPCM,MP3/MPEG2	Como o conteúdo de mídia é codificado e identificado para interoperabilidade entre dispositivos
Media Transport	HTTP	Como é que é transferido o conteúdo de mídia
Media Management	Arquitetura UPnP	Como o conteúdo mídia é identificado, gerido e distribuído
Discover & Control	Arquitetura UPnP	Como os dispositivos se auto-configuram, descobrem-se e se controlam mutuamente
IP Networking	IPv4	Como os dispositivos se conectam entre si e comunicam
Connectivity	Ethernet/Wi-Fi/Bluetooth	Como os dispositivos realizam a conexão à rede

As camadas Connectivity e IP Networking são responsáveis por definirem como é que os dispositivos se conectam na rede local (*wi-fi/ethernet*) e como é que realizam as comunicações (IPv4ou IPv6).

Tanto a camada Media Management como a Discover & Control seguem a arquitetura do *UPnP*. Estas são responsáveis pela descoberta, configuração automática e controlo dos dispositivos.

A camada Media Transport define os mecanismos de transporte ou transferência do conteúdo pela rede, utilizando para tal *Hypertext Transfer Protocol* (*HTTP*) [23] e *Real-time Transport Protocol* (*RTP*) [1].

Na camada Media Formats são definidos os conjuntos de formatos de média, tanto opcionais como obrigatórios, para cada diferente tipo de dispositivo e para cada uma das três classes de multimédia: imagem, som e vídeo.

Por fim, a camada Link Protection, assegura que o conteúdo comercial está protegido da pirataria e da redistribuição ilegal. Permite que seja efetuada uma distribuição dos conteúdos de uma forma segura numa rede doméstica, preservando os direitos de cópias dos fornecedores ou dos donos dos conteúdos. [21]

2.1.3 *DAAP*

A tecnologia Digital Audio Access Protocol (DAAP) [24][25] foi introduzida pela Apple na plataforma iTunes na versão 4.0. Um servidor DAAP é um servidor HTTP especializado que executa duas ações distintas:

- Envia uma lista de conteúdo para o cliente;
- Efetua uma *stream* do conteúdo áudio solicitado pelo cliente.

Todos os pedidos são realizados através de pedidos GET do protocolo HTTP e usa o *gzip* para comprimir os conteúdos. Porém, o servidor é capaz de enviar as respostas sem que exista esta compressão. Esta codificação é usada apenas para que o conteúdo leve menos tempo a ser transferido. O servidor responde então com um tipo de dados *application/x-dmap-tagged mime-type* que é facilmente obtido e convertido no formato XML por parte do cliente.

À semelhança do DLNA que se apoia no UPnP, o DAAP/iTunes utiliza o serviço *ZeroConf*, mais conhecido como *Bonjour*, para anúncio e descoberta de servidores numa rede doméstica. Utiliza aeste protocolo sobre a camada de transporte e, por defeito, está em escuta na porta 3689[26].

Para além do DAAP, a Apple utiliza outra tecnologia para partilha de conteúdos multimédia, o Digital Photo Access Protocol (DPAP), que neste caso é utilizado por outro produto da empresa, o iPhoto. Ambos dependem de um protocolo de baixo nível, Digital Media Access Protocol (DMAP). De salientar, que para além do DAAP existe o Digital Audio Control Protocol (DACP), e ambos são responsáveis por controlarem os servidores e por tratarem da comunicação cliente-servidor.

Bonjour

Bonjour [27] é um conjunto de protocolos sobre a camada IP que usa conceitos de configuração zero de rede (*Zeroconf Networking*). O Bonjour, à semelhança do UPnP, tenta resolver o desafio de, numa rede local IP, de adicionar dispositivos e estabelecer uma conexão entre todos (quer sejam computadores, impressoras, etc) com um mínimo ou até mesmo zero configurações de rede. O utilizador deve conseguir descobrir os serviços existentes na sua rede local sem que para isso seja necessário possuir conhecimentos técnicos, como por exemplo, saber o endereço IP de cada dispositivo.

Assim sendo, o Bonjour trata de todas as configurações necessárias numa rede doméstica sem que o utilizador tenha de possuir conhecimentos para o fazer. Para tal, esta tecnologia propõe soluções para três áreas essenciais:

ADDRESSING (atribuição de endereços IP) – é conseguido através da função *self-assigned link-local addressing*. Para tal, é usado um intervalo de endereços que estão reservados apenas para redes locais (LAN). No IPv6 esta caraterística já se encontra incorporada no protocolo, mas não em IPv4. A solução adaptada neste protocolo é escolher aleatoriamente um endereço da gama disponibilizada e testar se o mesmo já se encontra atribuído. Caso o resultado seja negativo então pode adotar o endereço encontrado, caso contrário tem de aleatoriamente escolher um novo endereço;

NAMING (atribuição de nomes) – baseia-se no uso de *Multicast DNS* (mDNS), em que todos os pedidos DNS são enviados através de *multicast IP*. Desta forma, não é necessário o uso de um servidor de DNS global, visto que todos os pedidos são enviados para endereços *multicast* e cada dispositivo é responsável por anunciar as suas caraterísticas. Em computadores ou dispositivos iOS, o *Bonjour* inclui um *handler* para tratar das respostas aos pedidos mDNS. Este *handler* anuncia a disponibilidade do serviço pelo que todos os pedidos mDNS são direcionados para o IP e porta correspondentes;

SERVICE DISCOVERY (descoberta de serviços) – permite que as aplicações descubram quais os serviços existentes e para cada um manter uma lista dos nomes e portas que providenciam tal serviço. Os nomes devem ser resolvidos como mencionado anteriormente em *naming*. A lista associada a cada serviço permite que a mesma seja atualizada sem que sejam feitos grandes anúncios na rede. No *bonjour* é acrescentada a funcionalidade de *browsing*, em que, cada pedido mDNS é enviado para um certo tipo de serviço e todos os serviços que correspondem ao pedido respondem com os seus nomes. Deste processo resulta uma lista de nomes que é capaz de incluir o serviço requisitado.

Autenticação DAAP

Como já referido anteriormente, a tecnologia DAAP foi introduzida na versão 4 do iTunes. Nesta versão, a tecnologia DAAP possuía um sistema de autenticação o que implicava que apenas clientes com instâncias de iTunes se pudessem conectar entre si. Posteriormente, numa atualização do *software* iTunes a Apple modificou o seu sistema de autenticação utilizando um algoritmo de *hash*. Porém, ambas as versões sofreram *reverse engineering* o que permitiu que clientes que não possuíssem o aplicativo iTunes se conseguissem conectar aos servidores. [28]

Para reverter esta situação, na versão 7 do iTunes, a Apple acrescentou um cabeçalho adicional, *Cliente-DAAP-Validation*, que é necessário para estabelecer uma conexão. Este cabeçalho não afeta os servidores DAAP, no entanto afeta todos os clientes que não estejam preparados para utilizar este cabeçalho (mesmo versões anteriores ao iTunes 7). No processo de autenticação, a *hash* que é gerada é determinada através do cabeçalho adicional. Este tipo de autenticação ainda não foi alvo de *reverse engineering*. [29]

2.2 APLICAÇÕES DE *streaming* DE MÚSICA

Hoje em dia existem inúmeras plataformas de *streaming* que nos permitem aceder a milhares de músicas. De seguida, apresenta-se uma breve descrição sobre quais as principais aplicações que existem na Internet em que é possível reproduzir conteúdo musical e/ou controlar uma coleção.

2.2.1 *Spotify*

O Spotify [30] é um serviço de *streaming* de música, vídeo e *podcasts* que foi oficialmente lançado em 2008 na Europa, chegando mais tarde aos EUA. Permite o acesso a milhões de músicas e outros conteúdos de artistas de todo o mundo. Podem ouvir-se álbuns inteiros, bem como listas de reprodução criadas pelo próprio utilizador ou até mesmo pelo *staff* do Spotify (i.e., um processo automático). Está disponível numa panóplia de dispositivos: computadores, *smartphones*, colunas, *tablets*, televisões e até mesmo em carros. Possui uma grande interoperabilidade devido ao Spotify Connect [31] que permite ouvir facilmente a música desejada numa coluna ou televisão, por exemplo, utilizando a aplicação Spotify como controlador. Para tal, todos os dispositivos devem de estar na mesma rede local.

É um serviço *freemium*, i.e., apresenta uma componente básica inteiramente grátis, e um serviço de subscrição *premium*. Na componente gratuita os utilizadores consomem publicidade um pouco por toda a aplicação, desde *banners* durante a reprodução até a interrupções no fluxo entre duas músicas, que ocorre periodicamente de trinta em trinta minutos. Já os utilizadores que pagam pelo serviço *premium* não têm publicidade, podendo assim ouvir as suas músicas sem qualquer interrupção e efetuar o *download* para ouvir as músicas desejadas em modo *offline*, entre outras vantagens. De seguida, enumera-se uma série de caraterísticas básicas (i.e., da componente grátis) contempladas pelo Spotify [30]:

- Permite que o utilizador selecione o que deseja ouvir através de ferramentas de procura e pesquisa;
- Apresenta recomendações de acordo com o gosto musical dos utilizadores;
- Permite a construção de coleções musicais, através da criação de lista de reprodução;
- Permite a partilha de canções entre amigos, tendo acesso ao conteúdo que reproduziram, bem como o de artistas e celebridades (desde que este tipo de informação seja declarado como público);
- Permite que o utilizador crie a sua própria estação de rádio.

No entanto, quem subscrever o serviço *premium* tem uma séria de caraterísticas adicionais, como [32]:

- O não consumo obrigatório de publicidade;
- Saltar músicas de forma ilimitada (na versão simples os utilizadores apenas podem saltar seis músicas a cada hora);
- Ouvir música *offline*, desde que se tenha efetuado o *donwload* das músicas desejadas previamente;
- Melhor qualidade de som. Na versão grátis as músicas são reproduzidas a uma taxa de 160kbit/s, ao passo que na versão *premium* são reproduzidas até 320kbit/s [33].

Na altura de escrita deste documento,, o Spotify oferece a possibilidade de o utilizador experimentar o serviço *premium* por um prazo de três meses. Passado este período experimental, a subscrição custa cerca de 7€/mês para o plano individual. Um estudante subscreve por 3,50€/mês e existe um plano familiar que permite associar até 6 contas, pelo valor de 11€/mês.

Arquitetura

Nos primeiros anos do Spotify, devido à incapacidade da rede e também dos servidores, existiam três estratégias diferentes para obter a música [34]:

CACHING – sempre que um cliente ouvia uma música esta era guardada temporariamente em cache no dispositivo local do utilizador.. Assim, se fosse selecionada novamente a música estaria diretamente disponível para o utilizador sem ser necessário efetuar qualquer pedido pela rede. O espaço disponibilizado era configurável, indo desde 1GB até aos 100GBs;

SERVIDOR SPOTIFY – quando a música selecionada não estava presente na cache era necessário efetuar um pedido para a obter. A música era dividida em partes (em *chunks* à semelhança do BitTorrent). Os primeiros 15 segundos de cada música eram sempre requisitados ao servidor.

REDE *OVERLAY PEER-TO-PEER* – como o utilizador só obteve uma parte da música através dos servidores, era necessário obter as restantes partes para que a música ficasse completa. Para as adquirir, numa primeira fase, os pedidos eram redirecionados para outros utilizadores, vizinhos na rede P2P. Caso, o *match* fosse conseguido iniciavam uma troca de conteúdo musical, senão o pedido teria de ser redirecionado para os servidores do Spotify.

De salientar que o processo de *caching* tanto servia para o cliente utilizar localmente, como podia ser disponibilizado para a rede P2P. Desta forma, o cliente servia de servidor, em relação a outros clientes, para disponibilizar partes de música que possuísse localmente.

A rede P2P apenas estava disponível para as versões *desktop*, uma vez que os *smartphones* ainda não possuíam grande poder computacional e o foco era o reprodutor multimédia nos computadores.

Porém, em meados de 2014 o *Spotify* deixou de utilizar a rede P2P [35]. Tomaram esta decisão porque já possuíam servidores com grande capacidade e espalhados por todo o mundo para conseguir responder a todos os pedidos de forma eficiente e a rede P2P causava um *overhead* na rede devido à sua manutenção [36]. De salientar que no caso de *streaming* de vídeo, que pode consumir bastante mais largura de banda, o uso de redes P2P ainda pode fazer sentido.

Uma das caraterísticas mais importantes do Spotify é o controlo das músicas para um sistema de recomendação, que o serviço pretende que seja o mais fiável possível, de acordo com os gostos musicais do utilizador. A tecnologia utilizada variou ao longo dos anos [37]. Sempre que um utilizador começa a ouvir uma música ou coloca um "adoro" ou então começa a seguir um artista, despoleta um evento. Esse evento é apanhado numa *streaming pipeline*, através da tecnologia Pub/Sub da Google [38], e a informação fica disponível em tempo real para poder ser processada. Nestas *streaming pipelines* podem ser efetuadas diversas operações para manipular e/ou adquirir os dados, através de *aggregates*, *grouBy*, etc. As recomendações são então criadas com base no comportamento por parte utilizador na aplicação.

Para agrupar as coleções musicais, o Spotify utiliza um modelo de *machine learning* que permite encontrar uma representação vetorial para as palavras, Word2Vec, sendo que palavras no mesmo espaço significam que têm mais em comum, o mesmo se passa com as músicas, i.e., músicas agrupadas no mesmo espaço significa que têm um género semelhante. Desta forma, intersetam as músicas disponíveis com as ouvidas por parte do utilizador, conseguindo assim descobrir novas músicas que possam agradar por serem do mesmo estilo. Este conjunto de recomendações, que designam de *shelves*, é então guardada numa base de dados chave-valor, que tipicamente é a BigTable [39]. Sempre que o utilizador inicia a aplicação, então é efetuado um pedido para obter as recomendações para o mesmo. A figura 4, apresenta o fluxo deste processo.

2.2.2 *Outras Aplicações*

De seguida apresentam-se outras aplicações bem conhecidas que estão disponíveis na Internet e cujo objetivo é semelhante ao do Spotify.

APPLE MUSIC [40] – esta aplicação é bastante parecida com o Spotify. Aliás, estas duas aplicações são as dominadoras no mercado relativo ao *streaming* de músicas. Permite que o utilizador selecione qual a música que pretende ouvir, podendo criar uma lista de reprodução personalizada. Para além disto, também efetua recomendações aos

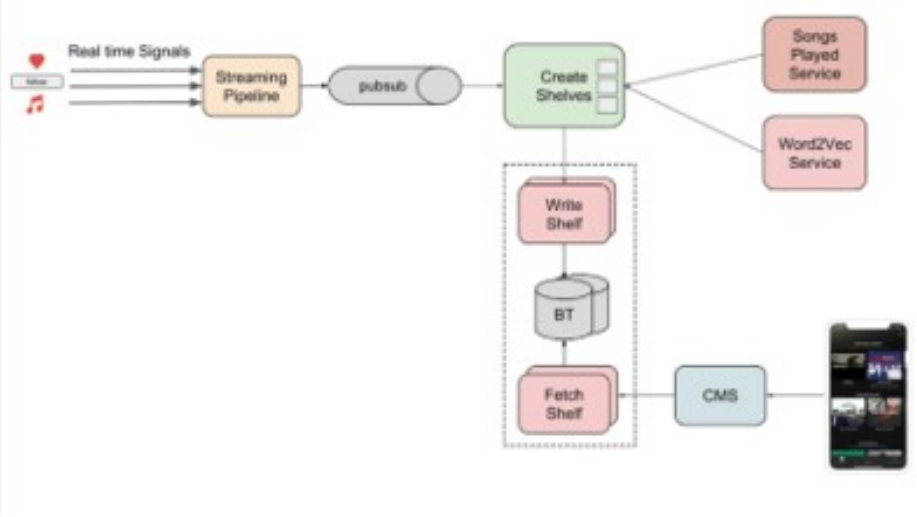


Figura 4: Arquitetura de Sistema de Recomendacoes do Spotify [37]

utilizadores, tentando indicar músicas que sejam as mais idênticas ao gosto musical dos mesmos. Permite ainda que o utilizador, em modo *offline*, consiga reproduzir as suas músicas (necessário efetuar um *download* da música, antecipadamente, e não apenas marcá-la como favorita). Uma vez que é uma aplicação produzida pela Apple possui compatibilidade com diversos dispositivos da marca, como por exemplo o Apple Watch, para escutar as músicas que deseja. Esta funcionalidade é conseguida através do AirPlay.

PANDORA [41] - foi desenvolvida pela Music Genome Project, que posteriormente foi adquirida pela Sirius XM Sataellite Radio [42]. É uma aplicação que apenas está disponível nos Estados Unidos da América, mas que possui uma grande popularidade face à qualidade e diversidade no catálogo das músicas e ainda as capacidades que disponibiliza aos utilizadores para encontrarem músicas semelhantes ao seu gosto musical. Os utilizadores são ainda capazes de dar um *feedback* sobre as músicas que a aplicação lhes sugeriu, indicando apenas se gostam ou não. Esta informação é depois utilizada para melhorar a sugestão de novas músicas por parte dos desenvolvedores. Esta aplicação possui versões para dispositivos móveis, como também para *web browser*.

YOUTUBE MUSIC & GOOGLE PLAY MUSIC [43] - estas aplicações complementam-se e foram as duas produzidas pela gigante em tecnologias, a Google. Inicialmente, a aplicação para reproduzir músicas da Google era o Google Play Music, que permitia efetuar o *upload* e o *streaming* de músicas. Porém, esta aplicação não teve tanto sucesso porque não disponibilizava tantas funcionalidades, como por exemplo o Spotify, o que fez com que a Google lançasse uma nova aplicação para rivalizar com as restantes aplicações e gradualmente vai tornando obsoleta a mais antiga [44]. Assim, em junção com o

YouTube (que foi adquirido pela Google) foi criado o Youtube Music, que permite não só ouvir, como ver um vídeo associado. Ou seja, não só faz *streaming* de áudio como também *streaming* de vídeo para reproduzir o videoclipe associado à música. A grande vantagem desta aplicação é a sugestão das músicas em tempo real. Como por exemplo, quando uma pessoa vai ao ginásio sugere músicas que são capazes de aumentar o ritmo cardíaco para que o exercício efetuado seja mais eficiente [45].

Existem mais aplicações cujas funcionalidades são parecidas às apresentadas anteriormente, como a Deezer[46], Tidal [47] ou a Amazon Music [48]. Todas elas servem para reproduzir músicas remotamente na Internet utilizando o *streaming*. Músicas essas que não estão na posse do utilizador e para que ele as possa ouvir é bombardeado com publicidade (nas aplicações *freemium*, como o Spotify), ou então, terá de subscrever um pacote *premium* que lhe remove a publicidade e lhe permite acrescentar uma série de funcionalidades, como poder reproduzir as músicas mesmo sem acesso à Internet. A tabela 2 apresenta um comparativo entre estas aplicações.

Tabela 2: Comparação Aplicações de *Streaming* Musical (em julho 2021)

Aplicação	Spotify	Apple Music	Yotube Music	Pandora	Amazon Music	Deezer	Tidal
Grátis	✓	✗	✓	✓	✗	✓	✗
Preço (€/mês)	6.99	6.99	6.99	6.99	6.99	6.99	6.99
Plano Família/Estudante	✓	✓	✓	✓	✓	só família	✓
Músicas	+50M	+60M	Indeterminado	2M	+50M	+35M	+60M
Modo Offline[1]	✓	✓	✓	✓	✓	✓	✓
Partilha de Músicas	✓	✓	✓	✓	✓	✗	✓
Podcasts	✓	✓	✗	✓	✗	✗	✗
Android	✓	✓	✓	✓	✓	✓	✓
iOS	✓	✓	✓	✓	✓	✓	✓
Disponível Portugal	✓	✓	✓	✗	✓	✓	✓
Utilizadores Pagantes	+100M	+60M	+15M	+10M	+30M	+7M	+3M

Como é possível analisar pela tabela 2 as caraterísticas de todas as aplicações são semelhantes, sendo que a escolha do utilizador pode ser muito influenciada pela interface ou recomendação de um conhecido ou então se pretende ouvir *podcasts* (que é a caraterística mais diferenciadora).

Existem ainda outras aplicações de *streaming* de música com um conceito diferente. Um desses casos é a MixCloud [49], fundada por quatro ex-universitários que se conheceram num espetáculo de rádio. O conceito desta aplicação assenta na divulgação de conteúdo próprio onde diversas pessoas, como DJs ou apresentadores de rádio, possam efetuar o *upload* do seu conteúdo para que este seja divulgado para todos os utilizadores. Todos os direitos são reservados a quem publica o conteúdo, desde que não seja fraudulento. Para conseguir manter os custos de desenvolvimento os criadores tiveram de lançar um pacote *premium* [50] que custa cerca de 7 USD mensais (julho 2021) e que permite aos utilizadores ouvirem músicas ilimitadas.

1 Apenas com *premium*

Relembra-se ainda o iTunes [51], um serviço original criado pela Apple. Hoje em dia, o iTunes enquadrasse mais num reprodutor multimédia com uma particularidade, possui integrado uma loja, iTunes Store, com uma diversidade de conteúdos (músicas, *podcasts*, vídeos, etc) que podem ser adquiridos pelo utilizador. Assim que um utilizador compre um conteúdo na iTunes Store, então uma cópia da música passa a ser propriedade do próprio podendo esta ser transferida e armazenada localmente. O iTunes utiliza o DAAP para poder encontrar e conectar outros dispositivos locais para *streaming* de conteúdo, sendo ele próprio o controlador do fluxo de reprodução.

2.3 APLICAÇÕES *cloud* DE *streaming* DE MÚSICA

Um serviço de *cloud* de *streaming* de música, não só permitem que um utilizador armazene a sua música privada num único local remoto da Internet, como também permite que o mesmo possa reproduzi-la através do *streaming*.

A diferença para as aplicações mencionadas na secção anterior é que uma aplicação de *streaming* de *cloud* permite que o utilizador armazena e aceda ao seu próprio conteúdo, ao invés de ser apenas o conteúdo disponibilizado ou controlado pela aplicação de *streaming*.

Este tipo de aplicações existentes no mercado são uma opção melhor para aqueles utilizadores que possuem já uma vasta coleção privada, porém como é um serviço de *cloud*, e não apenas de *streaming*, tem custos associados para suportar maiores quantidades de armazenamento. Quanto maior a coleção privada do utilizador, maior será o custo para a poder aceder totalmente a partir destes serviços. A transferência e manutenção de grandes coleções privadas nestes serviços de *cloud* pode ser pouco amigável e é preciso algum conhecimento técnico para fazer a sua gestão eficientemente.

Na tabela 3 estão apresentadas as caraterísticas das principais aplicações de *cloud* de *streaming* de música presentes no mercado. Constata-se que a Google Play Music [43] é a aplicação com um maior custo para o utilizador e que em termos de qualidade de *streaming* a VOX Cloud Music [52] é quem oferece um melhor serviço, permitindo efetuar a reprodução do áudio com a qualidade original. Tanto a My Music Cloud [53] como a VOX Cloud Music permitem efetuar um carregamento ilimitado de músicas, sendo que apenas a segunda é que apresenta compatibilidade com o formato FLAC. O iTnues Match [54] é um serviço extra que a Apple oferece para complementar o iTunes.

As aplicações Google Play Music e Amazon Music também foram mencionadas na secção anterior isto porque são aplicações cujo o seu foco é o *streaming* de música, mas também apresentam este serviço de *streaming* de música a partir da *cloud*.

A DoubleTwist [55] também permite efetuar o *streaming* de música a partir da *cloud*, mas com a particularidade de não oferecerem um serviço de armazenamento em *cloud*. Nesta aplicação o utilizador tem de efetuar uma configuração para estabelecer uma ligação a um

Tabela 3: Comparação Aplicações *Cloud* de *Streaming* Música (julho 2021)

Aplicação	Google Play Music	Amazon Music	iTunes Match	My Music Cloud	VOX Cloud Music
Grátis	✓	✓	✓	✓	✗
Preço serviço *premium* ($/ano)	120	25	25	40	50
Nº de músicas	até 35M	até 250K	até 100K	Ilimitado (serviço *premium*)	Ilimitado (serviço *premium*)
Qualidade máxima (kbps)	320	256	256	320	original
Sincronização com outros dispositivos	✓	✓	✓	✓	✓
Suporta FLAC	✗	✗	✗	✗	✓
Android	✓	✓	✓	✓	✓
iOS	✓	✓	✓	✓	✓
Disponível Portugal	✓	✓	✓	✓	✓

dos seguintes serviços: Dropbox, OneDrive ou Google Drive. A aplicação é totalmente gratuita, e permite realizar o *streaming* da coleção privada, desde que a mesma esteja armazenada numa das *clouds* mencionadas.

Associado ao serviço DoubleTwist temos outras aplicações, das quais podemos destacar o CloudPlayer [56] para efetuar *stream*, procura e descarregamento de músicas guardadas na *cloud* e ainda o Classic Player [57] e Sync [58] para partilhar o conteúdo musical numa rede local, quer através de AirPlay como de DLNA. De salientar que o Classic Player é um reprodutor de multimédia, que permite ao utilizador controlar toda a sua coleção, ao passo que o Sync é um *software* que pode ser instalado noutros reprodutores, como o Windows Media Player, criado pela Microsoft.

2.4 TRABALHOS RELACIONADOS

A discussão deste tema das plataformas e das tecnologias de *streaming* áudio é amplamente dominado por duas vertentes distintas:

- Uma em que se enquadram as soluções comerciais desenvolvidas e disponíveis e que já foram abordadas nas secções anteriores. Neste contexto, importa discutir as funcionalidades disponíveis e, se possível e quando a informação está disponível, as arquiteturas em que os componentes se organizam, e as situações específicas a que estas soluções melhor se adequam;

- Uma outra em que se enquadram as tecnologias, mecanismos e protocolos base que suportam as soluções comerciais referidas. Neste contexto, podem ainda abordar-se as tecnologias aplicacionais mais específicas como o UPnP, o DLNA e o DAAP e, eventualmente, mecanismos e protocolos de mais baixo nível como o HTTP, o RTP ou o Multicast IP, ou mecanismos de codificação de fontes áudio.

Conforme já foi referido no primeiro capítulo, o objetivo principal desta dissertação era enquadrar o trabalho de investigação e desenvolvimento precisamente na primeira vertente, ou seja, na definição duma arquitetura que permita suportar com maior facilidade, e duma forma mais eficiente, os requisitos funcionais específicos dum grupo de utilizadores mais

restrito, quando comparado com as soluções amplamente usadas na atualidade. Ou seja, não foi relevante, no contexto desta dissertação, o estudo muito detalhado das tecnologias, mecanismos e protocolos base referidos na segunda vertente, pelo menos aquelas que estivessem abaixo das tecnologias aplicacionais.

Uma consequência deste enquadramento foi a dificuldade em encontrar referências puramente científicas que se relacionassem com relevância com os trabalhos desenvolvidos nesta dissertação. No entanto, merece destaque um trabalho científico de investigação e desenvolvimento de cariz académico desenvolvido na Universidade do Minho e do qual resultou a definição de uma plataforma para *streaming* de áudio numa rede local através da utilização exclusiva de tecnologias e protocolos abertos que não estão associados a nenhuma propriedade empresarial [59]. Nesta arquitetura, trocou-se a utilização do UPnP e do DLNA por uma tecnologia inteiramente baseada em Simple Network Management Protocol (SNMP) [60].

Acesso a Conteúdo Musical Privado Via Protocolo SNMP

O O principal objetivo do trabalho descrito em [59] é a gestão e reprodução de conteúdo musical numa rede doméstica, assegurando que se usa *software open source*.

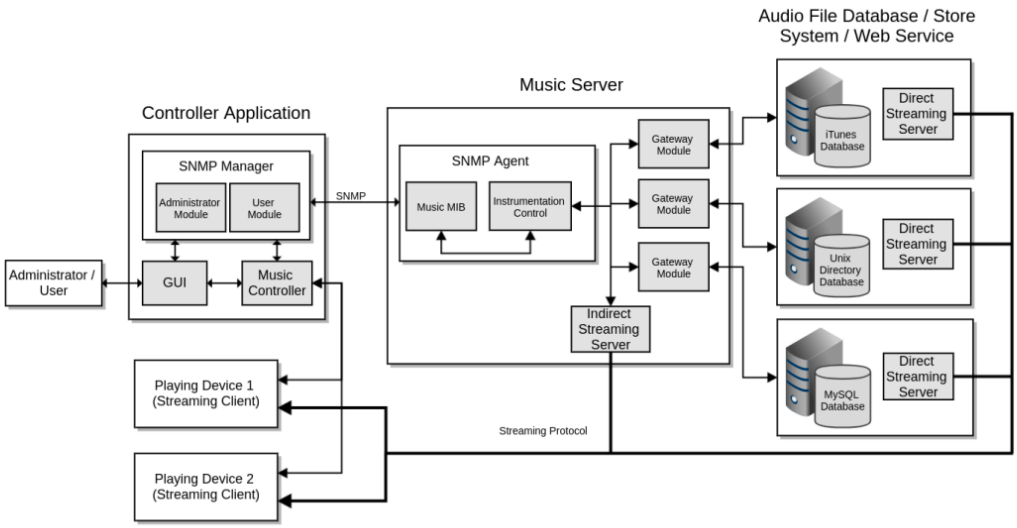


Figura 5: Arquitetura de *streaming* áudio baseado em SNMP [59]

Na figura 5 representa a arquitetura do sistema que possui 4 componentes principais:

MUSIC SERVER: guarda a informação de meta dados, que vai ser acedida pelo Controller Application. Contêm um agente SNMP que implementa uma base de dados, neste caso uma MIB, em que os objetos representam meta-informação relativa ao conteúdo

guardado nos *audio file databases*. Este agente possui um módulo Instrumental Control de alto nível que implementa métodos capazes de transformar a semântica abstrata dos objetos da MIB de músicas em operações genéricas de nível intermédio para aceder e controlar a música. Para além disto, o Music Server é capaz de iniciar uma *stream* caso o sistema de dados não seja capaz de o fazer;

AUDIO FILE DATABASE: sistema de base de dados que contém a coleção musical. Para cada base de dados é necessário um modulo *gateway* para que cada tecnologia de armazenamento das músicas seja suportada pelo Music Server. Este *gateway* fará a ligação entre as operações genéricas do Music Server e as tecnologias suportadas pelas Audio File Databases;

CONTROLLER APPLICATION: gere a informação no Music Server, que inicialmente é carregada pelos módulos Instrumental Control e *gateway*. A informação inclui os meta-dados associada à base de dados específica do utilizador. Permite que um utilizador comum interaja com todas as funcionalidades do sistema, como por exemplo, pesquisar por músicas através de filtros, pausar ou parar músicas, verificar os dispositivos disponíveis e reproduzir uma música diferente ou independente em cada um, entre outras. Permite ainda ao administrador de sistema gerir o seu catálogo, como adicionar/remover músicas, definir operações para os *gateways*, dispositivos e servidores, entre outras;

PLAYING DEVICES: os dispositivos que reproduzem a música para o utilizador.

De referir que, caso seja o Music Server a iniciar o *streaming* das músicas, não é guardada qualquer cópia do conteúdo, garantindo que os direitos de autor não são violados durante o fluxo de reprodução.

Seguem-se algumas das funcionalidades disponibilizadas pelo sistema:

- A descoberta de dispositivos é efetuada através de mensagens "Hello"que são enviadas frequentemente. Cada mensagem tem informação sobre o endereço IP, porta da aplicação e ainda o tipo de componente. Caso os componentes estejam registados num ou mais servidores DNS então estas mensagens de "Hello"não precisam de ser enviadas;
- A possibilidade de pesquisa de conteúdo musical tendo em conta um critério especificado pelo utilizador;
- Suporte para vários formatos de áudio (MP3, AAC, WAV, FLAC e OGG).

3

SOLUÇÃO PROPOSTA

Conforme foi referido nos capítulos anteriores, o principal objetivo deste projeto foi a definição de um sistema que permita efetuar o *streaming* de conteúdo musical armazenado num servidor privado, garantindo-se desta forma que o utilizador consegue reproduzir músicas da sua coleção privada em qualquer lugar do mundo desde que tenha acesso à Internet com uma largura de banda e uma fiabilidade mínima. Mais ainda, é importante ressalvar que os servidores nas redes locais dos utilizadores continuem a salvo de ataques de utilizadores maliciosos na Internet.

No capítulo 2 foram abordadas algumas tecnologias e protocolos que permitem implementar sistemas de *streaming* de conteúdo áudio (como o UPnP, DLNA e DAAP), mas o seu contexto de aplicação, sem adaptações, é restrito a redes locais pelo que não resolvem o problema do *streaming* remoto através da Internet.

Por outro lado, todas as plataformas de *streaming* remoto existentes no mercado, também revistas no capítulo anterior, como por exemplo o *Spotify*, no entanto, a premissa de funcionamento das mesmas não cumpre os requisitos desta dissertação, ou seja, estas plataformas mais conhecidas não permitem cumprir um requisito importante de poder aceder às coleções musicais armazenadas e geridas em servidores privados de redes privadas dos utilizadores.

Outras aplicações menos conhecidas como a VOX Cloud Music ou DoubleTwist disponibilizam serviços de armazenamento remoto em *cloud* das coleções privadas dos utilizadores, permitindo depois o *streaming* entre os servidores da *cloud* e os reprodutores dos utilizadores em qualquer parte do mundo com acesso à Internet. No entanto, estes sistemas obrigam a um processo moroso de transferência das coleções privadas para a *cloud*, para além de não garantirem uma manutenção das coleções dos utilizadores que deixem de ter uma subscrição do serviço.

3.1 ARQUITETURA

O novo tipo de plataforma tem que poder ser implementada utilizando apenas tecnologias, mecanismos e protocolos abertos (*open-source*) e de uso completamente gratuito. Isto pressupõe que qualquer pessoa é capaz de aceder à tecnologia desenvolvida e correspondente

documentação, podendo efetuar modificações que satisfaçam requisitos particulares. Desta forma incentiva-se a comunidade a participar na melhoria do sistema a longo prazo, sem que a experiência de utilização seja afetada. É ainda possível que outros criadores de *software* acrescentem novos módulos e funcionalidades ao sistema original.

Como é apresentado na figura 6, o modelo proposto tem os seguintes componentes principais:

MUSIC SERVICE - gere toda a informação presente no sistema. Responsável por efetuar a ligação entre o cliente e a coleção musical requisitada, caso possua permissão para aceder à mesma. Se necessário, é capaz de efetuar o *stream* do conteúdo digital facultado pelo cliente;

LOCAL AGENT - responsável por aceder ao conteúdo musical privado gerido pelo utilizador. Todas as comunicações são efetuadas com o Music Service, exceto quando se está a realizar *streaming* do conteúdo musical diretamente com o cliente;

CLIENT APP - responsável pela reprodução do conteúdo multimédia. É o componente com uma maior interação com o utilizador final, permitindo-lhe visualizar todas as coleções musicais e, caso o utilizador tenha permissões, reproduzir na totalidade cada música selecionada.

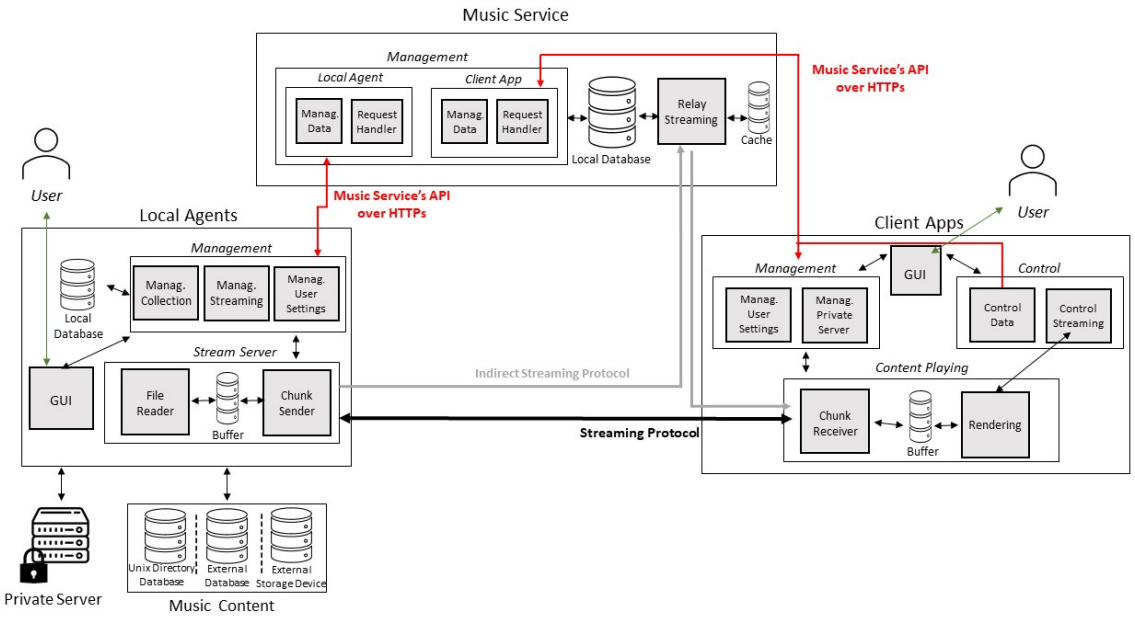


Figura 6: Arquitetura do Novo Tipo de Plataforma de *Streaming* Musical

Tanto a componente Client App como a componente Local Agent possuem interação com o utilizador através da GUI respetiva. Na primeira o cliente tem a capacidade de

controlar a reprodução musical (selecionar uma música especifica ou pausar a reprodução, por exemplo), *Control Streaming*, desde que tenha permissões de acesso ao repositório musical, que é realizado pelo Control Data.

Caso aceda ao servidor privado com permissões de administrador, o utilizador deverá ter a capacidade de gerir a coleção musical, como por exemplo, alterar as *passwords* de acesso ou indisponibilizar um conteúdo, através da implementação do módulo Management Private Server. O utilizador da Client App tem ainda a possibilidade de alterar os seus dados pessoais, sendo que esta gestão é realizada através do módulo Management User Settings.

A reprodução do conteúdo musical no Client App está assegurada pelo módulo Content Playing. Este é formado por dois sub-módulos principais: o Chunk Receiver, responsável por receber os diferentes segmentos da música e enviar as mensagens de controlo de *streaming*; e o Rendering que tem como função reproduzir adequadamente e sem interrupções os diferentes segmentos que estão armazenados no Buffer musical.

No Local Agent, a interação com o utilizador é mais limitada, quando comparada com a interação de um utilizador com o Client App. Inicialmente este tem a possibilidade de definir o local de armazenamento onde está localizado o conteúdo a ser disponibilizado e ainda a configuração do servidor de *stream*, como por exemplo, a definição do número de clientes que podem aceder ao conteúdo em simultâneo ou então se permite que no Music Service seja implementada uma cache temporária dos conteúdo musicais.

O módulo Management é responsável por gerir estes e outros parâmetros de configuração, sendo que o sub-módulo Management Streaming está relacionado com as definições de *streaming*, ao passo que o sub-módulo Management User Settings está relacionado com as definições gerais do servidor privado do utilizador. Além disso, através do Management Collection é efetuada uma gestão da coleção musical.

No módulo Stream Server é onde acedido e enviado o conteúdo da coleção requerido. É realizado em vários segmentos através dos módulos File Reader e Chunk Sender, respetivamente, tendo em conta as mensagens de controlo de *streaming*.

O component Local Agent tem de estar instalado no servidor do utilizador para se que possa realizar o processo de *streaming*, sendo que o conteúdo pode ser acedido a partir de uma diretoria (ou conjunto de diretorias) dum sistema de ficheiros, de uma base de dados externa ou então de um outro tipo de armazenamento gerido e mantido em servidores privados na rede local do utilizador.

3.1.1 *Music Service*

A componente Music Service é responsável por efetuar a ponte de ligação entre as outras duas componentes principais, Local Agent e Client App. Toda a informação sobre a configuração dos diferentes Local Agents é mantida e gerida na base de dados, através do

módulo, Management - Local Agent, que pode ainda conter dados referentes a pequenos excertos das músicas, para quando o cliente efetuar o pedido de *preview* de uma música. Por uma questão de eficiência e de armazenamento, o excerto não deve ultrapassar os 20 segundos e deve ser mantida uma qualidade baixa de áudio. Esta medida justifica-se pelo facto de o Local Agent ser um servidor privado com menores capacidades em comparação com um servidor de um grande serviço, como por exemplo o Spotify. Devido a esta natural limitação, o servidor privado não tem capacidade para responder fielmente a muitos pedidos em simultâneo, pelo que qualquer medida no sentido de diminuir o número de pedidos a este componente, deve ser tomada. O facto de só se guardar apenas um pequeno excerto da música não se está a violar os *copyrights* já que a mesma não é guardada na sua totalidade. Adicionalmente, também é de esperar que o número de acessos em simultâneo a um mesmo Local Agent dum utilizador privado nunca seja elevado porque a coleção é privada e, em condições normais, apenas o utilizador dono da coleção estará a aceder ao Local Agent. em qualquer dos casos, por uma questão de segurança, o número de acesos em simultâneo permitidos pelos Local Agents pode serr parametrizado pelo utilizador responsável.

Por outro lado, o Music Service guarda informação sobre o utilizador, registando quais os repositórios a que o mesmo tem acesso além de toda a sua atividade (este assunto será abordado com mais detalhe adiante). O tratamento dos pedidos bem como a manutenção de toda a informação é gerida pelo módulo Management - Client App.

O Local Agent como o Music Service apresentam um comportamento semelhante ao de um servidor. O Music Service regista informação suficiente para responder aos pedidos dos utilizadores realizados através do Client App, deixando apenas que o Local Agent responda a pedidos de *streaming*. A quantidade máxima de ligações de *streaming* é definida por parte do dono do servidor privado, devendo estar de acordo com as capacidades da máquina.

Na comunicação entre o Local Agent e o Music Service o primeiro assume um papel de cliente ao passo que o segundo assume um papel de servidor, já que os pedidos são despoletados pelo Local Agent na procura de um serviço que é disponibilizado pelo Music Service. No entanto, quando o Music Service efetua o *relay* do *streaming* assume o papel de cliente, já que estará a efetuar pedidos ao Local Agent, que foram despoletados pelo utilizador a partir da componente Client App. Esta situação ocorre se não for possível estabelecer uma conexão direta entre o Local Agent e Client App, por exemplo, pelo bloqueio da *firewall* (em qualquer um dos componentes).). De realçar que estes problemas de acesso à rede privada no modo de *streaming* indireto podem ser ultrapassados mais facilmente adicionando regras de permissão de acesso à rede privada através do serviço bem conhecido do Music Server (em que, inclusivé, a utilização de certificados digitais é possível). Já no modo de *streaming* direto pode ser mais complicado resolver estes problemas da mesma forma porque os pontos de acesso onde estão os dispositivos que correm a Client

App podem estar também eles em redes privadas por detrás de sistemas NAT ou serem pontos acesso em redes móveis.

Refira-se que a arquitetura deve permitir também uma solução em que o Local Agent assume o papel de servidor virtual, i.e., na realidade o Local Agent é um cliente que vai verificando continuamente o estado dos pedidos de *streaming* no Music Service e quando os houver é o próprio Local Agent que inicia a conexão de transporte que permite a realização do *streaming*. Este modo de funcionamento resolve os problemas de segurança e de acesso referidos anteriormente sem necessidade de configurações com regras de exceção nem utilizar sistemas de autenticação mais complexos. Em contra-partida, esta solução tem o inconveniente de acrescentar um pequeno atraso na resposta a um pedido de *streaming*, ainda que neste tipo de serviço isso não acarrete uma degração significativa na experiência do utilizador.

A tarefa mencionada de *streaming* indireto é executada e gerida no módulo de Relay Streaming, sendo que é designado um armazenamento distinto da base de dados local para manter uma cache temporária dos ficheiros musicais transferidos.

Escalabilidade

Tanto no Client App como no Local Agent não é necessário tomar medidas ativas para aumentar a performance, ou evitar gargalos, uma vez que:

- O Cliente App está associado a um dispositivo privado do utilizador que deverá de ter a capacidade de reproduzir o áudio transferido, não tendo de responder a quaisquer pedidos;
- O Local Agent deverá ser um servidor privado limitado em termos de capacidades quando comparado com um servidor industrial, no entanto deve ser devidamente configurado por parte do utilizador responsável pelo mesmo, para que possa responder satisfatoriamente aos pedidos solicitados.

O Music Service é o componente central que recebe todos os pedidos de acesso por parte de todos os Client App (com exceção dos de *streaming*), realiza uma monitorização dos mesmos, e ainda efetua a comunicação de configuração e de ligação com todos os Local Agent. Logo, como é natural para os servidores que implementam estes tipos de serviços na Internet, o Music Service tem de ser o componente com maior capacidade, tanto ao nível de poder computacional como ao nível da largura de banda disponível, para conseguir responder a todos os pedidos que lhe serão efetuados.

Para tal, a hospedagem deste serviço na *cloud* é uma boa solução para resolver os problemas de escalabilidade. Esta instalação permite uma maior elasticidade, quer ao nível de armazenamento quer ao nível de poder computacional, já que existem serviços de *cloud* que

reservam recursos de acordo com o número de pedidos ao sistema, tomando as medidas necessárias para sincronizar os diversos pedidos e manter a informação consistente.

Modelo da Base de Dados

Como referido anteriormente, o Music Service é responsável por responder à maioria dos pedidos, bem como controlar e manter a informação referente aos diferentes repositórios (Local Agent) e aos diferentes utilizadores e respetivas Client Apps, quando ativas. Na figura 7 é apresentado um modelo da base de dados que descreve como se armazena a informação.

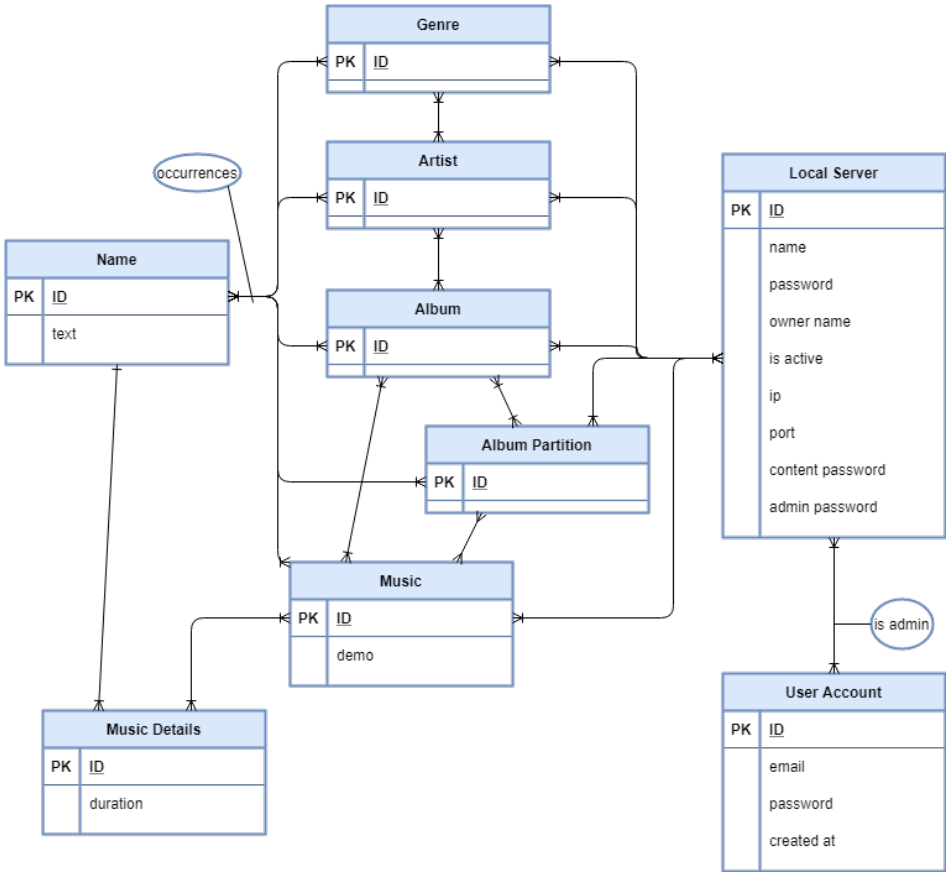


Figura 7: Modelo de Informação da Base de Dados do Music Service

A entidade Local Server representa os servidores privados (ou repositórios) associados ao sistema. Possui um identificador único que é gerado automaticamente pelo sistema e ainda um nome, que deve ser único. Este último é definido previamente pelo utilizador e validado no momento do registo do seu servidor privado. Contém informação sobre o endereço de comunicação, IP e porta (Port). É ainda mantido um atributo referente à disponibilidade do servidor. Quando um servidor privado é iniciado envia de imediato uma mensagem de *hello* para informar o Music Service que se encontra ligado e que pode começar a receber pedidos.

Esta mensagem deve ser enviada periodicamente para mostrar que se encontra ativo. Caso o Music Service deixe de receber estas mensagens, então assume que o servidor se encontra inativo e marca-o como tal, impedindo assim que um Client App realize pedidos de *stream* para o servidor privado.

Para que um utilizador num Cliente App possa aceder ao conteúdo dos repositórios privados, necessita de realizar uma autenticação prévia (por exemplo, através de uma palavra-passe).

Todos os atributos mais sensíveis, como por exemplo, aqueles relacionados com palavras-chaves, não são armazenados diretamente na base de dados, mas sim uma *hash* que os representa e identifica.

Relacionado com os servidores privados, é também guardada a meta-informação dos conteúdo disponibilizados no mesmo. Esta será usada para responder mais rápida e eficientemente a pedidos dos utilizadores sobre quais os conteúdos presentes em cada repositório.

Desta forma, a meta-informação vai associada aos gêneros, artistas, álbuns, partições de álbuns e os ficheiros musicais, que estão respetivamente representados pelas tabelas Genre, Artist, Album, Album Partition e Music. Todos estes elementos têm em comum uma caraterística, o seu nome, que nunca deve ser utilizado para o identificar unicamente. Recorde-se que a definição da informação está ao cargo dos utilizadores, e diferentes utilizadores podem representar a mesma entidade de forma distinta.

Assim, o sistema de armazenamento tem de ser flexível ao ponto de permitir ter várias caraterísticas para uma mesma entidade. Daí ser necessário definir antes duas entidades:

NAME: que possui uma ligação de muitos para muitos (N:M) com as seguintes entidades musicais (gênero, artista, álbum e partição de álbum, o que significa, por exemplo, que um nome pode estar associado a vários artistas e que um mesmo artista tem associado vários nomes;

MUSIC DETAILS: que tem uma conexão de muitos para muitos (N:M) com a entidade Music, ou seja, associada a uma música podemos ter várias caraterísticas, como uma mesma caraterística pode estar associada a músicas distintas. Nas caraterísticas das músicas, por exemplo, também é incluído o seu tempo de duração.

O relacionamento entre as diferentes entidades do modelo de dados da meta-informação segue o esquema de hierarquia do conteúdo no repositório local:

- Um gênero está associado a vários artistas e um mesmo artista pode estar associado a vários gêneros. Resultando uma ligação N:M, entre as entidades Genre e Artist;
- Um artista possui vários álbuns e um mesmo álbum pode ser da autoria de mais do que um artista. Resultando uma ligação N:M, entre as entidades Artist e Album;

- Um álbum possui várias partes de álbuns ou então músicas, e vice-versa. Resultando em duas ligação N:M, entre as entidades Album e Album Partition/Music. Entre as partições dos álbuns e as músicas também existe uma conexão de muitos para muitos.

Para ser possível identificar o conteúdo presente num servidor privado então é necessário estabelecer uma ligação entre as entidades que representam os dados da meta-informação (Genre, Artist, Album, Album Partition e Music) e a entidade que representa o servidor local. Ora, associado a um servidor privado podem estar associados mais do que um elemento de uma determinada entidade musical, e na ligação contrária também se aplica a mesma regra. Desta forma, resulta uma ligação N:N entre as entidades musicais, mencionadas anteriormente, e a entidade Local Server.

Como atributos na entidade Music existe um designado de Demo, que está relacionado com um excerto de 20 segundos que servirá para o utilizador poder pré-visualizar a música, aquando na navegação dos conteúdos dos repositórios através do Client App.

A informação sobre os utilizadores é mantida através da tabela User. Nesta entidade é uniformizado o conceito de utilizador da aplicação móvel e o utilizador que é responsável por um servidor privado. Um utilizador pode ter acesso a mais do que um servidor privado, e um servidor privado pode ser acedido por mais do que um utilizador, resultando uma ligação N:M entre a entidade User e Local Server. Nesta ligação é mantido um atributo que permite identificar se aquele utilizador é dono do repositório.

Devido às novas políticas do Regulamento Geral para a Proteção de Dados deve de ser apenas guardada a informação mínima necessária sobre um utilizador para o registar e identificar no sistema.

3.2 TRATAMENTO DE INFORMAÇÃO DUPLICADA

O Music Service é responsável por manter um registo de toda a meta-informação referente aos Local Agent de forma a providenciar uma resposta resposta o mais célere possível ao Client App sobre o conteúdo presente em cada repositório.

A meta-informação armazenada no Music Service não foi inicialmente definida pelo sistema, mas sim pelo utilizador que a detém, isto porque estes dados são obtidos do Local Agent correspondente sem que seja efetuada qualquer alteração ou manipulação do estado original dos mesmos.

O problema é os utilizadores podem ter formas distintas de representar a mesma informação. Por exemplo, enquanto uns definem um nome de um artista como sendo "Michael Jackson", outros podem defini-lo como sendo "Michael J". Em apesar de serem nomes distintos os dois podem referir-se ao mesmo artista.

Para além das caraterísticas distintas que caracterizam uma entidade, é necessário ter em atenção que dois servidores privados distintos podem possuir o mesmo conteúdo, ainda que

parcialmente. Isto é, nada impede que no servidor privado A esteja disponível um álbum que também está disponível no servidor privado B ou C.

O sistema do Music Service deve ser capaz de identificar que o conteúdo apresentado é o mesmo e evitar que a mesma informação esteja armazenada em duplicado.

A mesma lógica é aplicada ao Local Agent durante a extração da informação, porém a informação foi definida apenas por um utilizador, mas uma mesma música pode estar associada a dois álbuns distintos, pelo que a informação sobre a mesma deve de ser mantida apenas num local.

3.3 STREAMING

Por uma questão de eficiência, a maioria dos pedidos efetuados pelo Client App são tratados e respondidos pelo Music Service. No entanto, como o conteúdo musical só é diretamente acedido pelo Local Agent, obriga a que este seja o componente responsável para efetuar o *streaming* do mesmo. Isto implica que todos os pedidos de *streaming* sejam respondidos pelo Local Agent correspondente, sendo que a ligação deve de ser despoletada pelo Client App.

Por razões alheias aos diferentes componentes pode não ser possível realizar uma comunicação direta entre o Local Agent e o Client App (restrições de *firewall*, por exemplo). Para estes casos o Music Service possui um serviço de *streaming* indireto (Relay Streaming) que recebe os pacotes do Local Agent e redireciona-os para o dispositivo (Client App) correspondente. Nesta situação, o Client App, ao invés de efetuar o pedido para o Local Agent, envia-o para o Music Service e este redireciona-o para o servidor privado correspondente, assumindo o papel de cliente nesta última ligação. Quando o modo de *streaming* indireto normal também não é adequado então deve usar-se o *streaming* indireto mas em que o Local Agent assume o papel de servidor virtual.

Para aumentar a escalabilidade diminuindo o tempo de resposta nas funções aos utilizadores do serviço, o Music Service armazena temporariamente os ficheiros musicais. Desta forma, quando é realizada uma *streaming* indireta, caso a música se encontra na cache do Music Service não é necessário efetuar uma comunicação ao Local Agent uma vez que é o Music Service a responder diretamente ao cliente. Se o utilizador ainda tiver preocupações com direitos de autor (apesar do armazenamento em cache ser temporário) é possível o utilizador configurar o acesso ao serviço não permitindo o uso da estratégia de *caching* no acesso à sua coleção privada.

Uma vez que se trata de se transferir conteúdo no formato de áudio, não é necessário um protocolo complexo para efetuar o *streaming*, à semelhança do *RTP/RTCP* ou *HLS* que garantem a sincronização do áudio e vídeo. Desta forma, pode ser transferido utilizando simples *sockets TCP*.

O *Transmission Control Protocol* (*TCP*) possui mecanismos que controlam a ordem de entrega e o controlo de fluxo dos dados. Assim, quando o conteúdo for dividido em diferentes *chunks* associados a um intervalo da música e enviados por ordem temporal, é assegurado, pelo TCP, que a ordem temporal será a mesma na chegada ao cliente, pelo que a música é reproduzida de forma correta. Com o controlo de fluxo dos dados do TCP é garantido que, enquanto existe uma conexão à Internet, nenhum pacote é perdido sem conhecimento dos módulos aplicacionais e que é efetuado uma reprodução total do conteúdo musical. Porém, oo TCP pode adicionar um atraso na *stream* devido aos *handshake* inicial e todo o *overhead* imposto para o controlo na transmissão, no entanto uma vez que não se está a efetuar uma *live stream*, como por exemplo a transmissão de um jogo de futebol, este atraso tende a ser insignificante na experiência do utilizador.

Do lado do Local Agent os passos para realizar o *streaming*, gerido pelo módulo Stream Server, são descritos de seguida:

1. Receção e tratamento do pedido para identificar o conteúdo pretendido e iniciar a transmissão;
2. Acesso ao conteúdo indicado e divisão do mesmo em diferentes *chunks*;
3. Envio de cada *chunk* em pacotes individuais.

Quando o Local Agent recebe um pedido para reproduzir uma música de um utilizador devidamente autorizado responde para confirmar que o pedido será atendido e informa para que porta é que o Client App deve efetuar os pedidos de controlo para a reprodução do conteúdo musical.

A divisão do conteúdo pode diferenciar consoante o formato de *encode* utilizado. Por exemplo, 10 segundos de música em *MP3*, com um *bitrate* de 128kbps, equivalem a 160KB, ao passo que o mesmo intervalo de tempo num formato WAV, a uma frequência de 44.1KHz (qualidade de CD), é de aproximadamente 2MB. Ou seja, em termos de espaço ocupado (ou tempo de transmissão) podem existir grandes discrepâncias entre os diferentes formatos de codificação áudio. Como no lado do Client, para controlar a reprodução é importante ter uma noção temporal, os *chunks* são divididos tendo em conta intervalos da música. A escolha do intervalo será influenciada pelo formato e qualidade do áudio a ser transferido, sendo que cada *chunk* pode conter desde 100milissegundos até 1segundo de música.

Em cada *chunk* também é enviada meta-informação, que contém um identificador único, bem como o intervalo temporal correspondente na música. De salientar que o Client App já possui *à priori* meta-informação sobre a música requerida (informação disponibilizada pelo Music Service), pelo que essa informação não é preciso ser transferida durante o *stream*, estando unicamente a ser transferido o conteúdo musical e a meta-informação correspondente, i.e., uma *tag* de identificação e um selo temporal.

No Client App o processamento da transferência via *streaming* é efetuado no módulo Content Playing:

1. Envio do pedido para reprodução da música e tratamento da resposta para identificar a porta de comunicação;
2. Controlo do fluxo de reprodução, com armazenamento num *buffer* do respetivo conteúdo;
3. Renderização adequada do conteúdo do *buffer*.

Como durante a transferência por *streaming* o conteúdo não fica disponível na sua totalidade instantaneamente, é necessário armazenar as diferentes partes num *buffer*. No entanto, a reprodução não deve ser iniciada logo que se receba o primeiro *chunk*, uma vez que o Client App pode ser um dispositivo móvel que momentaneamente pode perder a ligação. Se o conteúdo fosse logo reproduzido, em alguns momentos a transferência pode falhar e desta forma a reprodução musical iria parar.

Para contornar este problema os *chunks* são armazenados num *buffer* e só quando possuir um determinado intervalo de música, ou então, quando 50% do espaço de armazenamento do *buffer* estiver ocupado, é que a reprodução deve ser iniciada. Após o início da reprodução sempre que existir um espaço livre no *buffer* deve de ser efetuado imediatamente um pedido para obter o próximo intervalo da música. Desta forma, mesmo que a transferência falhe momentaneamente não deve afetar a experiência de reprodução do utilizador.

3.4 Outras Caraterísticas do Sistema

Uma das preocupações do sistema é a disponibilidade dos diferentes componentes. Em relação ao Client App não é necessário qualquer medida uma vez que é um cliente e é quem solicita os pedidos. Sobre o Music Service, quer esteja hospedado na Cloud ou então instalado numa máquina, estará sempre ligado e pronto a receber conexões, pelo que o problema de estar ligado à rede não se coloca, e quanto à disponibilidade em si, só é afetada nos casos de problemas de escalabilidade (demasiados utilizadores em simultâneo, demasiados pedidos de *streaming* indireto em simultâneo, etc.). Mas como já mencionado anteriormente, estes problemas podem ser resolvidos com a instalação deste componente na Cloud para suprir a crescente exponencial de pedidos que possam ser efetuados num determinado período de tempo.

Por outro lado, como o Local Agent é um servidor privado controlado por um utilizador externo ao sistema, neste caso o dono do repositório, pode ser desligado ou então perder a conexão à rede ficando assim indisponível para responder a quaisquer pedidos efetuados por um cliente.

De forma a saber-se se o repositório está disponível é enviada, periodicamente, uma mensagem de *Hello* para o Music Service. Sempre que este componente receber esta mensagem que o Local Agent se encontra disponível para responder aos pedidos dos diferentes utilizadores. Uma vez que as comunicações são efetuadas através da rede, algumas mensagens podem ser perdidas, pelo que é importante ter um conjunto de regras que definam o estado ativo ou inativo de um Local Agent, que podem ser as seguintes:

- Ultrapassagem de um *timeout* temporal ao fim da última mensagem recebida;
- Qualquer outra mensagem recebida, pode contar como um *Hello*.

Além disso, deve ser registado o estado da *streaming* apenas no dispositivo do utilizador que está a efetuar a reprodução do conteúdo, i.e., no Client App. Assim, sempre que a aplicação for iniciada não é necessário efetuar qualquer pedido para saber o estado da *streaming* anterior, já que a informação estará acessível diretamente. Estes dados são gravados em cache, pelo que têm um tempo de vida limitado. Se, ao consultar os dados relativos à *stream* anterior estes não existam (não contenham informação), significa que o *TTL* expirou e que não faz sentido continuar com a reprodução antiga.

3.5 Análise de Segurança

O objetivo do sistema é permitir o acesso a servidores privados remotamente, o que implica que seja efetuada uma ligação através da rede aos mesmos. No sentido de tentar proteger os diferentes componentes desenvolveu-se uma análise de ameaças, *Threat Modeling*. O principal objetivo é identificar as lacunas de segurança existentes no *software* no sentido de mitigá-las antes de começar a implementar um protótipo do sistema.

3.5.1 *Utilizadores*

A interação com utilizadores é efetuada em dois componentes, Local Agent e Client App. Sendo que na primeira é bastante limitada, sendo só efetuada uma configuração do repositório, na segunda a interação baseia-se na reprodução de áudio e na navegação entre páginas de repositórios e de músicas. No caso de ter permissões de administrador, o utilizador é ainda capaz de alterar as configurações de acesso ao servidor ou então editar o conteúdo a ser disponibilizado. Assim, podem existir três tipos diferentes de utilizadores.

CLIENT APP :

- User
  - *Normal User* (utilizador normal de um repositório);

  - *Super User* (administrador de um repositório).

LOCAL AGENT :

- *Admin* (administrador).

De realçar que na Client App estarão presentes dois níveis de autenticação. Inicialmente, um utilizador tem de se registar na aplicação, ou seja, tem de se registar no serviço para poder visualizar os diferentes repositórios disponíveis, assumindo o papel de User nesta componente. Para aceder ao conteúdo na sua totalidade de um servidor privado, então necessita de ter permissões de acesso, e para tal deverá disponibilizar informação de autenticação adequada. Desta forma assume um papel de Normal User em relação aquele repositório. Mas caso tenha permissões para gerir o repositório então assume um papel de *Super User*. Assim, a distinção do papel de cada utilizador é feita de repositório para repositório.

Associada a cada tipo de utilizador existirá um número de funcionalidades disponíveis na plataforma.

User

Como já referido acima é um User é autenticado na Client App que possui uma lista dos repositórios a que o mesmo tem acesso. As funcionalidades que este utilizador tem acesso são:

- Procura e consulta da lista de repositórios (poderão ser aplicados filtros para uma procura mais rápida);
- Procura e consulta do conteúdo existente em cada repositório (poderão ser aplicados filtros);
- Visualização dos detalhes referentes a cada música, como por exemplo, título, duração, autor, etc;
- Editar as preferências do aplicativo.

Normal User

Após a autenticação válida para aceder a um determinado repositório o utilizador ganha permissões de acesso ao conteúdo musical, sendo que as funcionalidades na aplicação são as seguintes:

- Reprodução da amostra de conteúdo;
- Reprodução do conteúdo musical na sua totalidade;
- Capacidade de parar, pausar, continuar e avançar na música que está a reproduzir.

Super User

Nos casos em que o utilizador efetue uma autenticação para administrar o repositório, para além de possuir todas as funcionalidades de um Normal User, possui uma lista de funcionalidades extras relativas à gestão do repositório:

- Configuração do acesso ao repositório;
- Gestão da coleção musical (adição ou remoção de conteúdo).

Administrador

Um Administrador interage com o Local Agent de forma a configurar as definições do mesmo, permitindo definir, por exemplo, quais os utilizadores que têm acesso ao repositório. De seguida apresentasse uma lista das suas funcionalidades.

- Definição da localização do conteúdo a ser disponibilizado;
- Inserção da informação de acesso ao conteúdo;
- Configuração das definições dos pedidos, como por exemplo, periodicidade de pedidos, etc.

3.5.2 *Fronteiras de Confiança do Sistema*

De forma a entender melhor as interações entre os diversos componentes e os riscos associados aos mesmo, desenhou-se um esquema de fronteiras de confiança (*trust bounderies*) que está representado na figura 8. Assim, é possível verificar que as comunicações entre os diversos componentes, uma vez que podem ser efetuadas através da WAN, estão fora das zonas de confiança do sistema. Contrastando com as comunicações internas dos componentes que são completamente controladas e conhecidas.

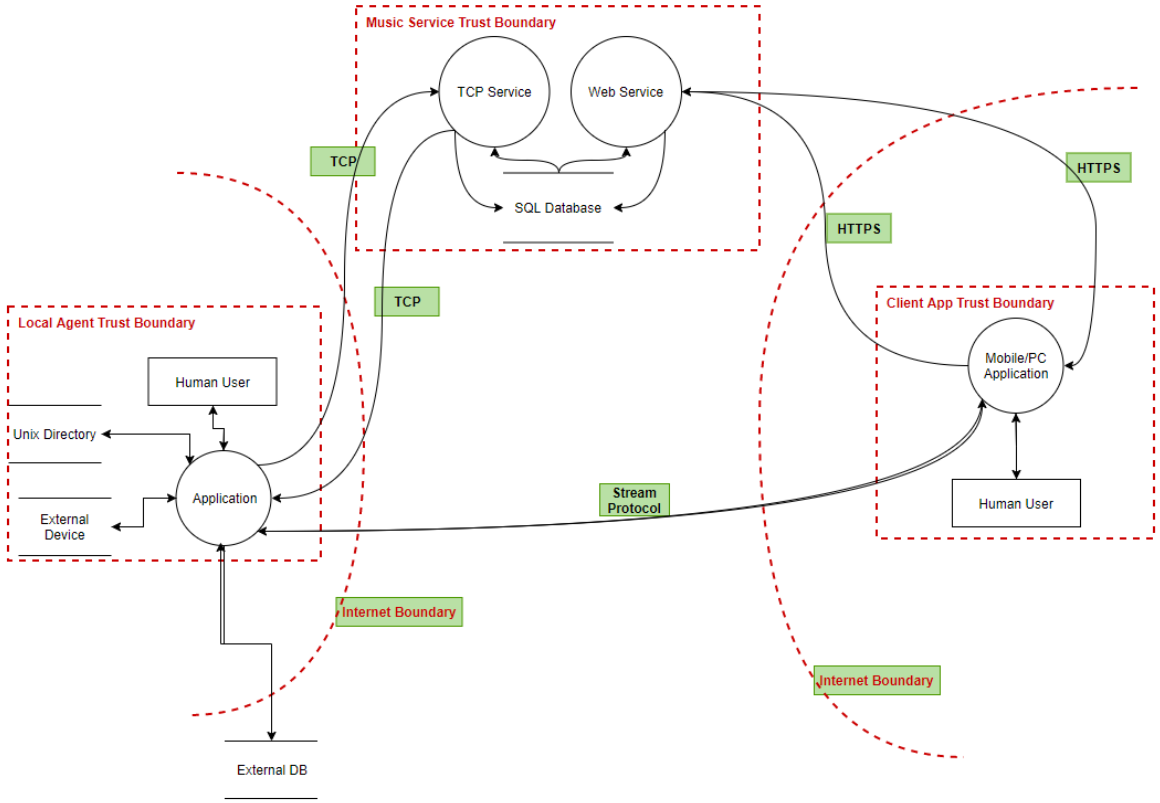


Figura 8: Fronteiras de Confiança

3.5.3 *STRIDE*

Para efetuar uma análise das ameaças a que o sistema está sujeito também se elaborou uma análise de risco, através da observação do sistema e das suas fronteiras de segurança evidenciadas na figura 8, com base nas categorias STRIDE [61]:

Spoofing

Spoofing refere-se ao acto de se fazer passar por outra pessoa ou declarar uma falsa identidade, quer em utilizadores como em processos. No sistema a ser desenvolvido um atacante pode tentar:

- Força bruta para descobrir a *password*;
- Obter as credenciais através de um falso *server*.

Por exemplo, estas ameaças podem ser minimizadas implementando um serviço de autenticação onde as *passwords* são cifradas, com uma chave segura que impeça a descoberta

da mesma por força bruta, e ainda um sistema de identificação por chaves digitais que permitem identificar inequivocamente tanto o cliente como o servidor.

Tampering

Tampering refere-se ao processo de modificação de dados ou processos. Um atacante, no nosso sistema, pode:

- Interceptar as mensagens enviadas entre componentes e modificar o conteúdo das mesmas;
- Alterar o destino das mensagens;
- Modificar os dados referentes ao conteúdo musical, no LocalAgent;
- Alterar a informação presente na base de dados do componente Music Servic;
- Injetar ataques através do interface normal da própria aplicação, conhecido como *SQL Injection*.

Quanto à intercepção das mensagens não há muito que possa ser feito, uma vez que as mesmas são realizadas sobre a Internet e é impossível controlar todos os pontos, o que não impede um atacante de as interceptar e modificar o seu conteúdo, enviando-as novamente para o destino correto. No entanto, se as mensagens forem assinadas digitalmente o destino é capaz de validar o conteúdo e verificar que o mesmo não é integro, descartando-o.

Para evitar injeções de ataque, como o *SQL Injection*, a medida normalmente adotada é o uso de procedimentos SQL, uma vez que não são executadas diretamente instruções SQL a partir do cliente. Quanto à proteção dos dados no Local Agent, pode ser instalado no servidor privado um programa para analisar eventuais intrusos de forma a impedir que os mesmos atuem, uma vez que nenhum componente do sistema tenta aceder diretamente à coleção musical. A mesma estratégia deve ser adotada para o Music Service, exceto se o mesmo for hospedado numa Cloud, nesse caso o próprio sistema já deverá possuir politicas de segurança para contornar estes problemas.

Repudiation

Repudiation é a negação da autoria de uma ação realizada ou de um evento que ocorreu. Como por exemplo, um atacante pode:

- Afirmar que não executou uma ação maliciosa (como alterar informação da base de dados do Music Service);
- Apagar informação dos *logs* para reduzir a sua atividade.

O não-repúdio é uma propriedade complicada de se garantir, uma vez que a pessoa pode sempre alegar que lhe roubaram as informações e que foi um outro que executou a atividade maliciosa. No entanto, a medida que é usualmente tomada, numa tentativa de se provar quem é que cometeu os atos, é o registo de *logs* de atividade no sistema para cada utilizador.

Information Disclosure

Information Disclosure acontece quando há uma divulgação inapropriada de informação, que pode ocorrer nos dados em trânsito, armazenados ou em processos. Um atacante tem a possibilidade de:

- Ler tráfico em texto não cifrado, durante a comunicação entre os diferentes componentes;
- Obter informações das bases de dados ou sistemas de ficheiros do Local Agent armazenados em ficheiros de texto não cifrado;
- Aceder a dados sensíveis guardados no Music Service.

A medida mais óbvia a ser para resolver este problema é a cifragem da informação sensível, como por exemplo, a informação de identificação e autenticação dos utilizadores. No entanto, não deve ser cifrada toda a informação porque, por exemplo, os dados referentes ao conteúdo musical de um repositório são acedidos publicamente para todos os utilizadores que possuem a aplicação. Deve ser feito uma escolha criteriosa dos dados a serem cifrados, uma vez que este processo diminui o desempenho do sistema.

Denial of Service

Denial of Service refere-e ao acto de provocar indisponibilidade, quer consciente ou inconscientemente, nos diversos componentes do sistema para os utilizadores que os querem usar devidamente. No sistema a ser desenvolvido um atacante pode tentar:

- Encher a capacidade das diferentes bases de dados, quer no Local Agent ou do Music Service;
- Inundar Music Service com uma quantidade enorme de pedidos em pouco tempo;
- Instalar no sistema um programa malicioso que tira partido do *hardware* para realizar uma atividade para a qual não foi desenhado, como minerar cripto-moedas.

Esta propriedade é a mais complicada de ser mitigada porque depende de inúmeros factores. No entanto, podem ser tomadas medidas que diminuam a probabilidade que isto aconteça, como aumentar os recursos sempre à medida que os pedidos aumentem, ou

bloquear um utilizador caso se suspeite de uma atividade ilícita. No caso do Local Agent deve ser evitada a partilha do seu endereço para este não receber pedidos externos ou então responder apenas a um número limitado de pedidos por intervalo de tempo, mas mesmo esta medida vai causar indisponibilidade, uma vez que se um utilizador lícito quiser iniciar um *straming* de música, não lhe vai ser permitido.

Quanto à base de dados, esta pode ser supervisionada. Quando estiver a ficar perto da sua capacidade máxima pode emitir-se um aviso e ser feita uma limpeza de informação que não seja importante. Em suma, não é possível mitigar por completo os problemas associados a este risco, sendo que são sempre precisas estratégias que permitam mascarar a indisponibilidade dos componentes.

Elevation of Privilege

Por fim, *Elevation of Privilege* acontece quando um utilizador ganha permissões de acesso para algo que não deveria de ter. No sistema a ser desenvolvido um atacante pode:

- Aceder a um repositório sem permissões, tanto a partir do Local Agent como da Client App.

Normalmente, estas ameaças estão muito ligadas com as de *Tampering*. Um bom sistema de autenticação que inclua um de *autorização* e controlo de níveis de acessos costuma ser suficiente para suprimir este tipo de ameaça.

4

IMPLEMENTAÇÃO E TESTES DO PROTÓTIPO

A solução proposta no capítulo anterior é complexa e possui inúmeros componentes. Desta forma, e de maneira a cumprir os prazos estabelecidos, foi estabelecido um protótipo de sistema que possui todas as funcionalidades básicas necessárias para a reprodução de conteúdo musical de coleções privadas.

Os componentes presentes no protótipo são o Local Agent, para ser instalado na máquina de acesso ao conteúdo musical privado, o Music Service, servidor central que gerência todos os Local Agents e permite o acesso dos clientes ao mesmo, e por fim, uma aplicação *mobile* para aceder e reproduzir o conteúdo. No anexo A será encontrado um manual que explica como funcionam a aplicação e o Local Agent, bem como as suas funcionalidades.

Na arquitetura da solução proposta é apresentado a possibilidade de o utilizador poder utilizar um computador ou então um dispositivo eletrônico com o sistema operativo Android ou iOS. No entanto, no protótipo apenas foi desenvolvida uma aplicação de utilizador compatível com um *smartphone* Android.

Relativamente à aplicação móvel possui as funcionalidades mais básicas implementadas, como listas os repositórios existentes bem como as músicas que possui, e ainda reproduzir a música com os seguintes controlos: pausar, retomar, avançar um intervalo de tempo de 30 segundos e recuar dez segundos. As opções de administração oferecidas pela aplicação móvel ao administrador contemplam apenas a visualização de quais os utilizadores que têm acesso aos seus repositórios.

A conceção do Local Agent também se encontra limitada na sua apresentação, uma vez que não possui uma interface para que o utilizador possa configurar o seu servidor. Para tal, é necessário a criação de um ficheiro de configuração que será carregado posteriormente no programa. Ainda no Local Agent, não foi desenvolvida a funcionalidade de servidor virtual, ou seja, um modo de funcionamento em que o Local Agent periodicamente verifica se existem novos pedidos de *streaming* através de pedidos realizados ao Music Service.

Outra diferença significativa entre a implementação inicial e a versão final do protótipo é o acesso ao conteúdo. Na arquitetura inicial existe a possibilidade de a coleção musical estar disponível no sistema operativo, numa base de dados externa, como uma NAS. Porém,

no protótipo apenas foi implementado o acesso a conteúdo que seja acessível fisicamente a parte da máquina onde o Local Agent se encontra instalado.

Neste capítulo será apresentada uma explicação técnica da conceção de cada componente (Local Agent, Music Service e Client App) do sistema bem como a forma como estão interligados. Será dado um maior foco ao processo de *streaming* e como o mesmo foi implementado bem como na questão de armazenamento da meta-informação de cada repositório no Music Service.

4.1 ARQUITETURA

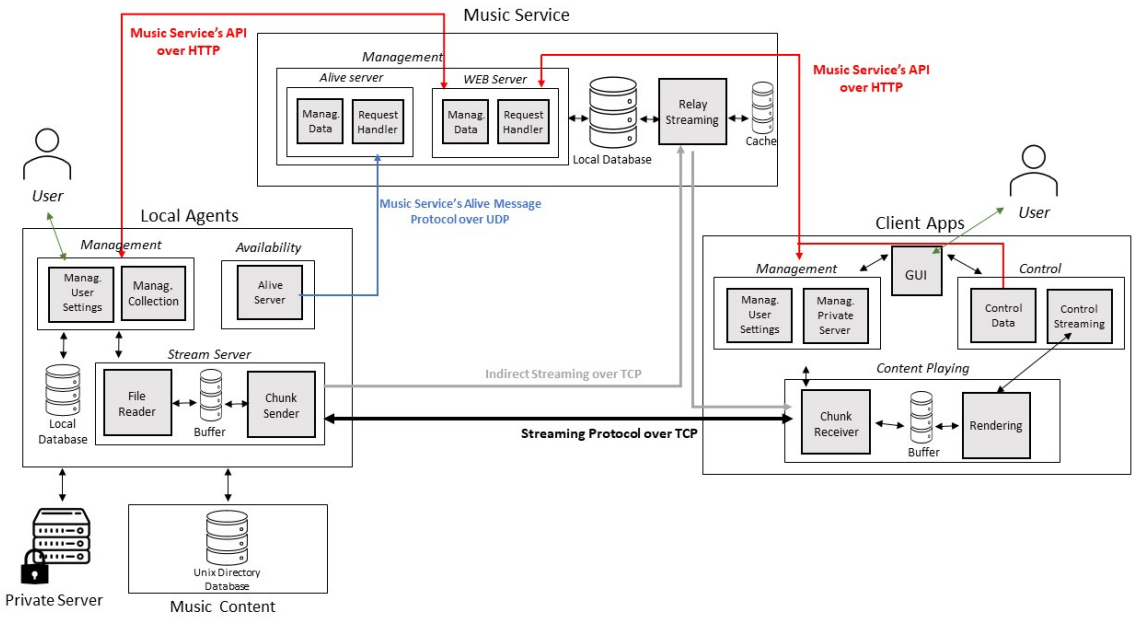


Figura 9: Arquitetura do Protótipo

Na figura 9 está representada a arquitetura do protótipo do sistema, que teve por base a arquitetura definida na secção 3.1. Existem três componentes principais: o Music Service, servidor central que gere todos os utilizadores e servidores privados para além de efetuar *streaming* (caso seja solicitado), o Local Agent, responsável pela gestão e distribuição de uma coleção musical privada, e por fim, o Client App, uma aplicação mobile para Android que permite navegar e reproduzir as diferentes coleções musicais.

O Music Service é o componente central do sistema e que serve de ponto de ligação para os outros dois componentes terminais, Local Agent e Client App. Este desdobrasse em três diferentes servidores:

SERVIDOR WEB: em escuta na porta 3000 com o endereço http://music-service-web-server.pt. Este servidor é responsável por atender todos os pedidos por parte do cliente da aplicação móvel;

SERVIDOR DE ALIVE: em escuta na porta 6001 sobre UDP onde recebe as mensagens de Alive dos diferentes Local Agent a indicar que os mesmos se encontram ligados e prontos para responder a pedidos de *streaming* por parte do Client App;

SERVIDOR DE *STREAMING*: está dividido em quatro *threads* cada uma em escuta numa porta distinta, desde a porta 62000 até à 62003 sobre TCP. Neste servidor é efetuado um redireccionamento do *streaming* caso não seja possível estabelecer uma ligação direta entre o Local Agent e o Client App. São utilizadas quatro portas para fazer um escalonamento do tráfego do Client App e Local Agent (duas portas distintas são utilizadas para cada um dos componentes) e ainda para diferenciar entre as mensagens de controlo e mensagens de dados (a questão do *streaming* bem como do *streaming* indireto será discutido ao detalhe na secção 2.2).

O servidor WEB foi construído através da tecnologia NodeJS, ao passo que os restantes foram construídos utilizando Java. Todos estes apresentam uma dependência para uma base de dados SQL, PostgresSQL, o local onde é armazenada toda a informação do sistema central (os dados dos utilizadores mais os dos servidores privados).

O Local Agent é um programa que deverá correr no *host* que possui acesso à coleção privada. Tal como nos diferentes servidores do Music Service, este também apresenta uma dependência para uma base de dados SQL, PostgresSQL, onde é armazenada meta-informação sobre a coleção privada para além de configurações.

O Local Agent é ainda responsável por transmitir os dados em *streaming* para o Client App (ou para o Music Service no caso de não ter sido possível estabelecer uma ligação direta, no entanto esta situação é transparente para o Local Agent). Desta forma, tem exposta a porta 61000 para escutar os pedidos de controlo de *streaming* por parte dos diferentes clientes e ainda expões uma série consecutiva de portas desde a 51000 até à 51100 para transmitir os dados de *streaming*. Em teoria, cada Local Agent consegue atender em simultâneo cem sessões de *streaming* diferentes.

Tanto o Music Service como o Local Agent estão distribuídos em *containers* Docker. No caso do Music Service permite uma maior flexibilidade para expandir para outros servidores. Sobre o Local Agent tem a vantagem de num *container* Docker ser compatível com vários sistemas operativos e que facilita a instalação do programa para aceder à coleção musical por parte de cada diferente dono. Para além das vantagens de distribuição das aplicações bem como da compatibilidade, facilita o ambiente de desenvolvimento do sistema para quem quiser contribuir, permitindo que mesmo utilizando tecnologias diversas o ambiente é igual para todos os desenvolvedores.

4.2 PROTOCOLO DE COMUNICAÇÃO

O sistema é constituído por vários componentes interligados construídos com diferentes tecnologias e que estão instalados em máquinas diferentes. Devido a este aspeto foi necessário definir um protocolo de comunicação que garantisse a serialização/deserialização das mensagens de igual forma qualquer que seja o dispositivo ou tecnologia utilizada.

Existe uma panóplia de métodos que permite uma serialização universal dos dados. Desde o *ANS1*, que é uma interface de descrição de mensagens independente da tecnologia utilizada, até *JSON*.

Para além da serialização dos dados é preciso ter em conta de que forma os mesmos serão transportados pela rede. Para informação sensível, como a questão da autenticação, e pedidos ao Web Server do Music Server o transporte é efetuado via HTTP. Na transferência de dados de *streaming* foi escolhido *sockets* TCP.

No caso da transmissão via HTTP a serialização é efetuada via JSON, uma vez que é cada vez mais utilizada numa comunicação *web-based* e está totalmente integrada com a tecnologia utilizada no Music Service. Para além disso, é compatível com muitas tecnologias para a vertente de desenvolvimento não sendo um fator para restringir a escolha.

Porém, JSON não é adequado para o tipo de dados que é transferido durante uma sessão de *streaming*, uma vez que são enviados dados de baixo nível, como por exemplo *bytes*. Devido à compatibilidade com inúmeras tecnologias e também pela sua performance, o método escolhido para a serialização dos dados foi o Protobuf. Possui suporte para vários tipos de dados, desde os mais simples (inclusive baixo nível, como o *byte*) a compostos (listas de elementos simples), e como é um método mais recente do que, por exemplo, o ASN1, a linguagem para definição das mensagens é *user friendly*, sendo muito facilmente lida e entendida por um humano, facilitando a compreensão das diferentes mensagens por parte do ser humano, sem afetar consideravelmente a performance na leitura por parte da máquina.

4.3 ORGANIZAÇÃO DO CONTEÚDO LOCAL

O conteúdo a ser disponibilizado encontrasse armazenado na máquina do Local Agent numa diretoria gerida pelo mesmo. Como os dados a serem armazenados dependem do utilizador existe uma norma que os mesmos devem seguir para organizar a sua coleção. Sem isto, a distribuição do conteúdo seria arbitrária dificultando a identificação das diferentes caraterísticas das músicas. Na imagem 10 está representada a hierarquia das diretorias.

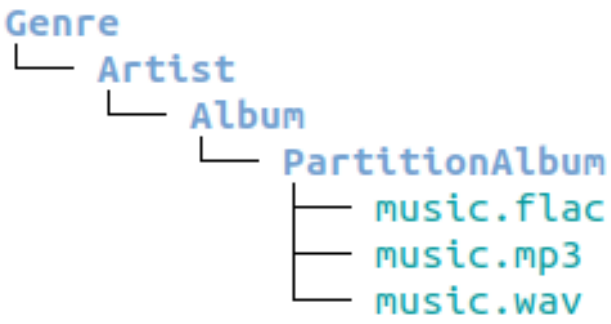


Figura 10: Hierarquia da Coleção

O utilizador define qual a pasta onde se encontra a sua coleção musical. Dentro desta, deve conter pastas que indicam o gênero musical. Dentro de cada pasta referente ao gênero, aparecem as pastas dos artistas ou compositores e qualquer ficheiro musical presente em qualquer uma destas pastas será ignorado.

A coleção de cada artista é composta por diferentes álbuns, pelo que o próximo nível da hierarquia é referente às obras ou álbuns. Chegado a este nível pode acontecer uma de duas coisas:

1. Existir um novo nível da hierarquia referente às partes ou andamentos ou atos dos álbuns/obras. Neste caso, só no nível a seguir da hierarquia é que aparecem os ficheiros musicais;

2. Está presente o conteúdo musical, o que indica que o álbum não é constituído por parte.

No entanto os utilizadores podem optar por não respeitar esta hierarquia desde que em cada pasta coloquem um ficheiro, de formato JSON, com a informação necessária para identificar o conteúdo existente. Este ficheiro pode ser utilizado como auxiliar para acrescentar informação sobre os ficheiros musicais, mesmo que o utilizador cumpra as normas hierárquicas definidas.

4.4 MUSIC SERVICE

O Music Service possui três serviços distintos:

WEB SERVER: responsável pela gestão e atendimento dos diversos utilizadores, Local Agent e clientes da aplicação *mobile*. A comunicação é efetuada via HTTP, sendo que a serialização das mensagens é realizada em JSON;

ALIVE SERVER: efetua a gestão da atividade dos diversos Local Agents;

STREAM SERVER: tem o papel de realizar o *relay* da *streaming* entre o Local Agent e o Client App, sempre que estes não conseguem estabelecer uma conexão direta. À semelhança do *stream* direto, os dados são transmitidos através de um protocolo definido por mensagens Protobuf, que são transmitidas via TCP. Possui ainda uma função de cache de músicas.

O Music Service é responsável pela gestão quer dos utilizadores da aplicação mobile (Client App) quer dos diferentes servidores privados (Local Agent) que se conectaram ao serviço. Para armazenar a informação referente a estas entidades é utilizada uma base de dados *SQL* que é comum a todos os serviços suportados pelo Music Service. Na imagem 11 está representado o esquema da base de dados para gerir toda a informação.

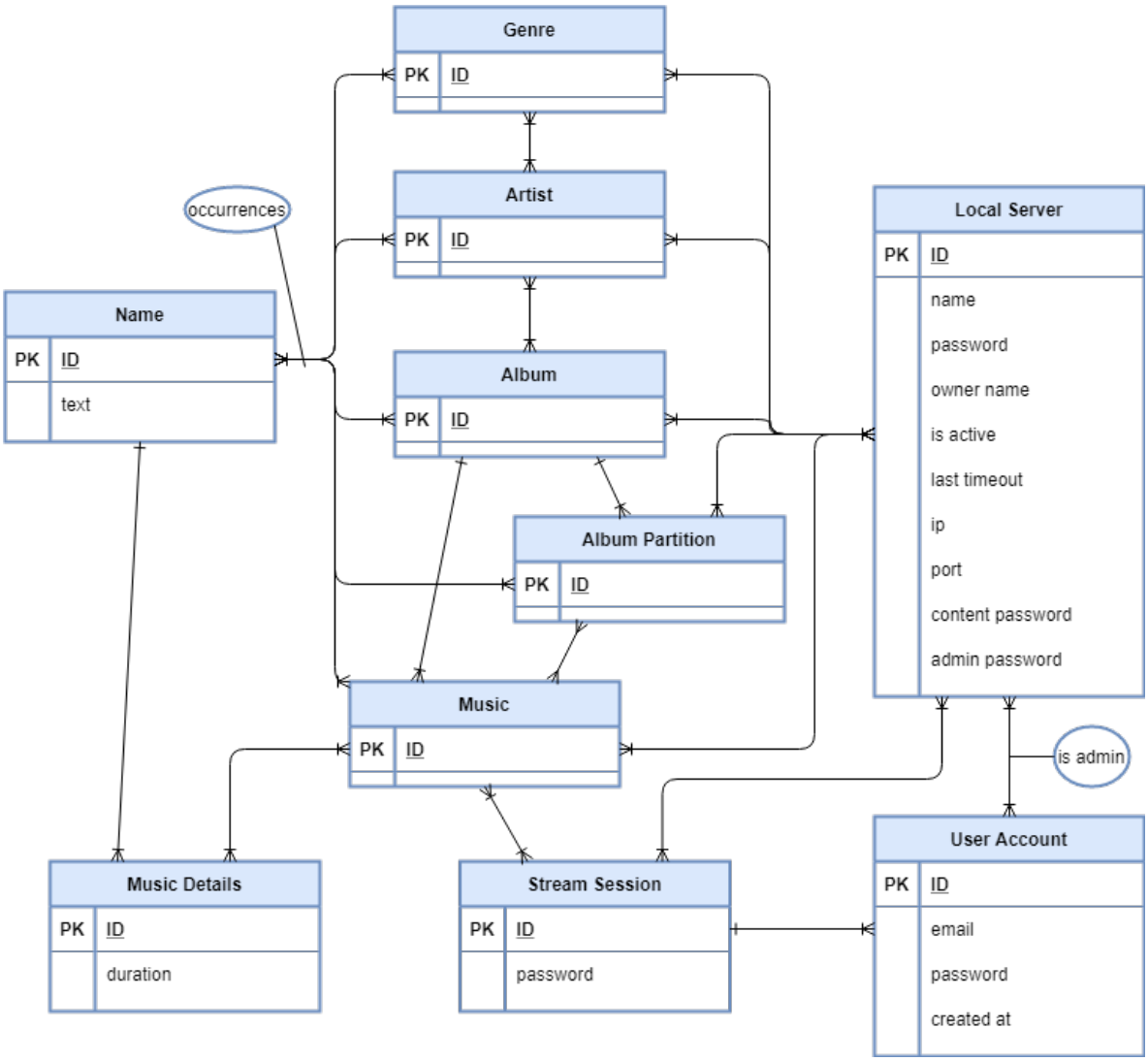


Figura 11: *Entity Relantionship Diagram* - Music Service

A componente Local Agent é representada pela entidade PrivateServer no esquema da base de dados. É guardada toda a informação pertinente sobre cada servidor privado:.

ID: identificador único gerado automaticamente;

NAME: nome único do utilizador do servidor privado, definido pelo dono do mesmo;

PASSWORD: palavra-chave de acesso ao servidor via Music Service;

OWNER NAME: nome do responsável pelo servidor privado;

ISACTIVE: permite identificar se um servidor privado está disponível;

LAST TIMEOUT: registo temporal da última atividade do servidor privado;

IP: endereço IPv4 da máquina;

PORT: porta aberta na máquina para escutar os pedidos referentes ao *streaming* do conteúdo;

CONTENT PASSWORD: palavra-chave para aceder ao conteúdo musical disponibilizado no repositório;

PASSWORD ADMIN: palavra-chave para aceder ao repositório privado com permissões de administrador.

Todas as palavras-chaves, como são dados sensíveis, não serão guardadas em *plain text*. Desta forma, a informação antes de ser inserida é cifrada no Music Service com um segredo que só o mesmo conhece.

Um utilizador do Client App, representado pela entidade User Account, tem a capacidade de aceder a diferentes repositórios, desde que tenha as permissões para tal. Enquanto a coleção musical de um servidor privado pode ser acedida por múltiplos utilizadores. Assim, formasse uma relação N:N entre as entidades User Account e PrivateServer.

Na entidade User Account são guardados os dados mínimos para que o mesmo possa aceder ao serviço e seja identificado no sistema, como o *email* e a *password*. Para cada utilizador é ainda gerado um identificador automático.

A entidade Music representa a música, neste caso, o ficheiro musical. A música em si, é um objeto complexo para ser representado uma vez que possui diversas caraterísticas, nome, duração, artista, álbum, gênero, etc. De modo a conseguir representar toda a complexidade associada a uma música, todos os componentes ligados foram repartidos em diferentes entidades:

MUSIC DETAILS - apresenta os detalhes simples associados diretamente à música, como a duração e o seu nome. Sendo que, como estamos a tratar de dados controlados por diferentes utilizadores, uma música pode ter associada diferentes detalhes e um detalhe pode estar a associado a mais do que uma música;

ALBUM - está associado diretamente à música e representa um conjunto de músicas que são comuns. Apesar de o álbum ter apenas como atributo um identificador único, é ainda representado por um nome, que pode variar devido à diversidade dos utilizadores. Neste caso uma música pertence a um e apenas um único álbum, ao passo que um álbum pode conter várias músicas;

ALBUM PARTITION - representa a partição de um álbum, desta forma está diretamente associado a um Album e à Music que pertence à respetiva partição;

ARTIST - representa o intérprete da música, porém não existe uma ligação direta à música, uma vez que um artista possui um conjunto de álbuns e os álbuns já estão associados a uma música, pelo que indiretamente é possível saber quais as músicas associadas ao artista. Para além do identificador único, para um artista estão associados vários nomes, pelas mesmas razões mencionadas nas entidades anteriores;

GENRE - representa o gênero musical da música. Utilizando uma estratégia similar ao Artist, esta entidade não se encontra ligado à Music, uma vez que o próprio artista possui um gênero.

Esta organização da informação permite aproximar o esquema do modelo de dados à hierarquia proposta para a disposição do conteúdo no servidor privado.

Por uma questão de simplicidade foi definido que uma música pertence a um e apenas um álbum. No entanto, existem situações em que um artista lança uma música associada a um álbum, e que posteriormente, vê essa música a ser inserida num álbum diferente constituído por músicas de artistas distintos. Desta forma, no mundo real a mesma música pode pertencer a dois álbuns, porém no sistema desenvolvido esta situação gera duas músicas distintas, mas que terão o mesmo artista associado.

4.4.1 *Web Server*

Como já referido anteriormente este servidor tem como principal função a gestão e ligação dos diferentes utilizadores da Client App. As ações que os utilizadores podem efetuar já foram mencionadas neste relatório, e é da responsabilidade deste servidor responder aos pedidos (com a exceção dos de *streaming* diretos) com a informação especifica.

A comunicação entre as duas componentes é através do protocolo HTTP. Por conseguinte, foi definido um sistema de rotas para requisição de informação sobre os servidores privados e coleção musical.

API de Dados

A API de dados é constituída por múltiplas rotas. As mesmas estão subdivididas por quatro categorias:

USERS: corresponde a todas as rotas direcionadas com a gestão de utilizadores. Permite que os mesmos, quer sejam um Local Agent ou usuário da aplicação *mobile*, realizem a autenticação e registo;

CONTENTS: referente ao conteúdo a ser armazenado no Music Service. Possui rotas para que o Local Agent possa inserir a meta-informação das músicas que disponibilizará e para que o cliente da aplicação *mobile* efetue uma pesquisa personalizada por nome;

STREAMS: apresenta as rotas relacionadas com as sessões de *streaming*. Permite que um utilizador da aplicação *mobile* crie uma nova sessão (ou atualize a atual) e que o Local Agent possa aceder às informações de uma sessão especificada no pedido (desde que o mesmo seja um dos elementos participantes da sessão);

SERVERS: corresponde a todas as rotas direcionadas com a apresentação do conteúdo disponibilizado pelos diferentes Local Agents.

As últimas três categorias são protegidas, ou seja, só um utilizador autenticado no sistema é que pode obter o conteúdo da mesma. Em rotas especificas é ainda aumentado o grau de proteção uma vez que é feita a verificação do tipo de utilizador que pretende obter os dados.

No caso das rotas que estão incluídas na categoria *servers* para além da validação da autenticação no servidor do Music Service é efetuada uma validação para verificar se possui acesso ao conteúdo disponibilizado pelo Local Agent. No caso de não possuir permissões de acesso não lhe é facultada a informação. O processo de autenticação encontrasse pormenorizado na secção 4.6.

4.4.2 *Alive Server*

Como quem providencia o conteúdo são servidores privados, cujo controlo da atividade do mesmo não depende de nenhum serviço do Music Service, foi necessário implementar uma estratégia que permita identificar se um servidor local se encontra ativo ou não. Desta forma, os servidores marcados como ativos podem efetuar o *stream* de conteúdo para os diferentes utilizadores que o requisitam.

No lado do Local Agent, é enviada, periodicamente, uma mensagem protocolar, Alive, num curto período de tempo, neste caso de cinco em cinco segundos. Para que um utilizador local se conectar ao Music Server, a primeira mensagem que envia é um Alive, para indicar que se encontra ativo e que pode receber pedidos de *streaming*.

No lado do Music Service é que a complexidade é maior, uma vez que tem de efetuar a gestão de todas estas mensagens de todos os Local Agents. De forma a efetuar um *parsing* eficiente deste tipo de mensagens, no Music Service, é aberta uma porta UDP, do conhecimento do Local Agent, onde apenas vai receber mensagens do tipo protocolar Alive, qualquer outro tipo de mensagem será ignorado. Com esta estratégia, impede que se receba todas as mensagens de controlo num único local, tendo de efetuar o *parsing* para cada uma para identificar que tipo é. Uma vez que se esperam mais mensagens do tipo Alive, faz todo o sentido isolar a receção das mesmas numa porta à parte das restantes.

Pelo facto de em simultâneo existir a possibilidade de estar um número considerável de servidores locais ativos em simultâneo, guardar a informação de cada um em memória não é plausível. Desta forma, toda a informação sobre a atividade dos diferentes servidores locais é guardada na base de dados. De seguida seguem detalhes dos dados a serem mantidos.

IS ACTIVE: um *bool* que indica se o servidor está ativo ou não;

LAST TIMEOUT: data, com dia e hora, da última mensagem de Alive recebida (quando o servidor passa de inativo para ativo, o valor é igual ao momento em que esta transição acontece);

TIMEOUT: tempo máximo, em segundos, que o servidor local demora a responder para ser dado como inativo.

De notar que o último atributo, nesta fase, será sempre fixo com o valor de cinco, referente à periodicidade do envio das mensagens do Local Agent. No entanto, numa fase posterior pode ser implementado um ajuste do tempo de resposta, consoante o histórico do tempo de envio do servidor local, tendo assim uma maior precisão sobre o estado de atividade do servidor local.

Após a falha de um *timeout* o servidor local é logo dado como inativo. Mesmo que o servidor local possa estar ligado, mas senão conseguir enviar mensagens ou receber então o mesmo deve de ser dado como inativo, uma vez que o mesmo está desligado em relação à rede. Assim que o Music Service volte a receber uma mensagem de Alive o servidor local, Local Agent, é dado como ativo novamente.

O uso de UDP ao invés de TCP justificasse pelo facto de serem mensagens muito simples e que são enviadas periodicamente, pelo que não se justifica o estabelecimento de uma ligação para o envio das mesmas. No entanto, com o uso de UDP não há garantia de entrega pelo que alguns pacotes podem ser descartados, o que pode levar a uma identificação errada de que o Local Agent se encontra inativo. No entanto, como as mensagens são enviadas num curto espaço de tempo este falso inativo pode ser rapidamente corrigido aquando da receção de uma nova mensagem de Alive.

Uma vez que não é feita uma monitorização ativa ao servidor local e que só de cinco em cinco segundos é que é possível avaliar se um servidor está ativo ou não, pode ocorrer falhas

no intervalo de tempo das monitorizações, pelo que neste espaço temporal o servidor local pode estar inativo e não ser detetado.

Esta situação afeta um cliente quando este pretende iniciar uma *stream* de conteúdo ou se atualmente já se encontra com sessão de *stream* ativa. Nestes casos o comportamento da Client App é efetuar uma tentativa de restabelecimento de ligação com o Client App e caso não consiga deve informar o Music Service e este tenta efetuar uma *stream* indireta, uma vez que o problema pode estar na ligação direta entre os dois componentes terminais, senão conseguir estabelecer a ligação ao servidor então o mesmo é dado como inativo.

Nesta situação a *stream* é interrompida e só quando o servidor voltar a ficar ativo é que é possível retomar o *streaming*. Nesta fase, nenhum tipo de informação é disponibilizado ao cliente, no entanto, numa versão futura, podem ser geradas notificações que indicam a atividade de um servidor local.

4.4.3 *Stream Server*

O Stream Server tem a função principal de assegurar o *streaming* de dados caso o cliente *mobile* não consiga estabelecer uma ligação diretamente com o Local Agent, desde que este último se encontre ativo.

O sistema implementado tem definido um protocolo de *streaming* no qual também vai ser utilizado pelo Stream Server, no entanto, no *streaming*, este possui um papel de servidor de *relay*, garantindo que efetua o redireccionamento para o destinatário pretendido. Os detalhes do processo de *streaming* serão apresentados na secção 4.7.

Para além desta função, o Stream Server apresenta uma cache de conteúdo, o que dá a possibilidade do Music Service responder a um pedido de *streaming* de um cliente sem que seja necessário efetuar uma ligação ao Local Agent.

No Music Service quando é recebido uma mensagem de *streaming* a mesma é analisada e é verificado se a música existe em cache. Se existir em cache então será sempre o Music Service a dar a resposta sem consultar o Local Agent.

No caso de a música ainda não existir em cache então é estabelecida uma conexão ao Local Agent e é efetuado um redireccionamento das mensagens. Essencialmente, no *streaming* existem dois canais, um para enviar mensagens de controlo e outro onde é enviado o conteúdo. Para armazenar a música, o Stream Server para além de efetuar um redireccionamento, guarda temporariamente os *bytes* que circulam pelo canal de dados, organizando-os em segmentos de 1 segundo.

Desta forma, para cada música há um conjunto de segmentos em que cada um possui a duração de um segundo. Assim, quando o cliente efetua um pedido por um segmento é logo acedido diretamente pelo Music Service e não precisa de estar sempre a ler todo o conteúdo.

Nesta implementação, só quando o servidor possui a música na totalidade é que começa a responder aos pedidos sem comunicar com o Local Agent. Nos casos, que só tenha sido transferida uma parte da música tem de ser sempre estabelecida uma conexão ao Local Agent.

Cada música em cache possui um *time to live* de trinta minutos e uma vez esgotado este tempo a música é apagada do servidor e um novo pedido para a mesma vai fazer com que seja estabelecida uma ligação ao Local Agent. Caso seja efetuado um pedido para obter a música, e esta se encontre em cache, então o seu *time to live* é atualizado para estar armazenada durante mais trinta minutos.

Nos casos em que é estabelecida uma ligação direta entre o Local Agent e o Music Service para a transmissão de conteúdo, como o Music Service não desempenha nenhum papel a música não pode ser armazenada em cache nem sequer os pedidos são respondidos pelo Music Service.

4.5 LOCAL AGENT

O Local Agent é o programa que lida diretamente com o conteúdo do servidor privado. É neste que está configurado qual o caminho para aceder à informação e que depois a agrega e guarda numa base de dados local.

O modelo de dados do Local Agent é em tudo semelhante ao que já foi apresentado para o Music Service, no entanto, possui mais informações para sobre o conteúdo musical. Esta informação está relacionada com o ficheiro em si, como o tamanho, *bytes* por segundo, tipo de ficheiro musical (mp3, flac ou wav), número de canais, taxa de amostragem (*sample rate*) e comprimento das amostras (*sample length*).

Para cada entidade possui ainda a informação de que se toda a sua informação já foi carregada para o Music Service e se existe alguma modificação recente no *filesystem*, referente ao caminho onde se encontra localizado o conteúdo, do servidor privado.

O Local Agent é sempre a primeira fonte para o *streaming* de dados. O processo de *streaming* será detalhado ao pormenor na secção 4.7. Mas num traço geral a realização de *streaming* de conteúdo acontece nos seguintes passos:

1. Receção e validação do pedido de reprodução do conteúdo por parte do cliente (quer seja um cliente de aplicação móvel ou o próprio Music Service);
2. Procura no sistema para encontrar a música requisitada pelo cliente;
3. Envio da meta-informação da sessão de *streaming*, como a porta para envio dos dados e informação sobre o conteúdo a ser transferido (tamanho, número de canais, etc);
4. Envio dos dados consoante as mensagens de controlo que são enviadas pelo cliente, até ao final da transferência do conteúdo ou por término de ligação por parte do cliente.

Apesar de o modelo de dados do Local Agent seguir a mesma estrutura que o modelo de dados do Music Service, o conteúdo presente não é igual ao conteúdo do Music Service, já que neste último é albergada toda a informação sobre os servidores privados.

Isto tem uma implicação de que os identificadores únicos para as entidades no Local Agent não são iguais (salvas raras coincidências) aos identificadores únicos das entidades no Music Service. Desta forma, a procura do conteúdo para a realização do *streaming* não pode ser feita através de identificadores. Para resolver esta situação é corrido um algoritmo de procura no lado do Local Agent, semelhante ao algoritmo de procura do Music Service, de forma a encontrar o conteúdo requisitado. Este algoritmo de procura será examinado ao pormenor na secção 4.8.

4.6 Controlo de Acesso e Autenticação

No sistema, como um todo, existem dois tipos de autenticação e ambas são dirigidas ao utilizador final, ou seja, pretendem validar a autenticidade do Client App quando é realizado um pedido para aceder aos serviços disponibilizados. De seguida, pormenorizasse as formas distintas de autenticação existentes.

4.6.1 *Music Service*

Para que um utilizador, a partir da aplicação, possa usufruir dos serviços disponibilizados tem de efetuar uma autenticação. Sem a realização deste passo o utilizador não tem capacidade para aceder a qualquer conteúdo providenciado pelo sistema. Desta forma, é possível inferir que não existe o utilizador *ghost* que acede livremente à aplicação e consome conteúdo sem qualquer tipo de identificação.

A validação da autenticidade dos diferentes utilizadores é efetuada por um simples formulário onde têm de colocar o *username*, neste caso o *email* de registo, e a respetiva *password*. Caso o utilizador não possua uma conta deve efetuar o registo providenciando informação básica sobre o mesmo, como o nome, gênero, data de nascimento e local de residência, para além do *email* e *password*.

O *email* é a caraterística que permite identificador unicamente um utilizador pelo que se, no momento do registo, o utilizador inserir um *email* que já esteja presente no sistema vai desencadear um erro, invalidando o registo e obrigando o utilizador a inserir outro *email* caso pretenda continuar.

Aquando do registo ou *login* de um utilizador do lado do servidor é gerado um *token*, em específico um *Json Web Token* (*JWT*), com algumas das informações pessoais do utilizador que é enviado no corpo da resposta, caso o pedido seja respondido com sucesso.

Todas as rotas disponibilizadas pela API do servidor *web* do Music Service estão protegidas, com exceção da rota de *login* e registo. Esta proteção, realizada através de um *middleware*, valida a veracidade do *token* que deve de ser incluído no *header* do pedido. Caso não esteja presente ou não seja válido é retornado um erro o que implica que o utilizador não tenha permissões para aceder ao conteúdo. Nestes casos, na Client App, o utilizador é redirecionado para a página de *login* para efetuar a autenticação.

Esta estratégia de autenticação é *sessionless*, ou seja, o servidor não guarda qualquer tipo de informação sobre o estado da sessão de autenticação dos utilizadores. Um ponto negativo é que não permite guardar informação sensível do utilizador no *token* uma vez que o servidor não possui total controlo do mesmo, pelo que só é possível extrair o *email* do mesmo. Mas, por outro lado, se a informação sobre a sessão fosse mantida do lado do servidor impediria que o mesmo escalasse. Tendo em vista estes dois pontos, é preferível que o servidor tenha uma maior capacidade de escalabilidade, mesmo que a informação presente no *token* tenha de ser mínima, mas suficiente para responder à maioria dos pedidos.

Uma vez que a informação sobre a sessão não é mantida do lado do servidor, neste caso no servidor *web* do Music Service, esta tarefa recai sobre a aplicação cliente, Client App. Desta forma, na aplicação é guardada alguma informação, no formato *key-value* (chave-valor), sobre o estado da sessão:

IS SIGN IN: é um valor verdadeiro ou falso, que permite identificar se existe um utilizador autenticado no momento, sendo que se o valor for verdadeiro indica que o utilizador está autenticado e se for falso indica que não há nenhum usuário com sessão aberta de momento na aplicação;

EMAIL: identifica o *email* do utilizador autenticado, caso não exista nenhuma este campo encontrasse vazio;

TOKEN: sequência de caracteres que correspondem ao *token* de validação de acesso ao conteúdo no servidor *web*, caso não exista nenhum utilizador autenticado este campo estará vazio.

4.6.2 *Servidores Locais*

Neste tipo de autenticação pretendesse validar que um utilizador da Client App, previamente autenticado, possui permissões para aceder ao conteúdo disponibilizado por um servidor local que se encontra ligado ao serviço.

Existem dois papeis distintos, de um utilizador normal e de administrador. Qualquer um pode ser desempenhado por um utilizador da Client App, mas a forma para os distinguir é com a *password* de acesso. Para se conectar a um servidor local, o utilizador só precisa de

indicar a *password* de acesso ao mesmo, sendo que se inserir a *password* de administrador tem a possibilidade de gerir o servidor privado remotamente.

A validação do acesso de um utilizador a um servidor privado é efetuada no Music Service. Sempre que um pedido, referente ao conteúdo de um servidor local, é efetuado ao Music Service é extraído o utilizador de origem e efetuada uma análise dos registos no sentido de verificar se o mesmo tem permissões para aceder ao conteúdo ou para efetuar a gestão do servidor local.

Na Client App é mantida uma lista com os nomes dos servidores privados que o utilizador tem acesso. No entanto, esta lista mantida do lado do cliente pode não estar atualizada, já que se existir uma mudança dos acessos aos servidores, os clientes não são notificados. Daí, a importância de ser sempre efetuada uma validação de acesso ao servidor local no Music Service.

Esta lista, guardada no lado do cliente, é atualizada sempre que é realizado um pedido ao Music Service para obter os servidores privados que estão associados ao utilizador autenticado. Com a lista dos servidores privados é possível diminuir os pedidos efetuados ao servidor privado a cerca dos servidores que o utilizador tem acesso.

Numa versão mais avançada, com o uso de notificações, pode ser feita uma atualização da lista de servidores locais sem que o cliente tenha de efetuar pedidos ao local server, e desta forma a atualização pode ser efetuada em *background*. Sempre que um utilizador se autenticar num Local Agent será gerado um *token*, mantido no lado do Client, que permita não só identificar o utilizador que realizada o pedido, como também o servidor que o mesmo pretende aceder, e se fizer correspondência com o nome do servidor privado requisitado então é disponibilizado o acesso ao mesmo.

4.7 STREAMING

No sistema desenvolvido é possível efetuar o *streaming* de duas maneiras. Primeira, e utilizada por defeito sempre que possível, a transmissão de dados é feita diretamente entre o Client App e o Local Agent, ao passo que na segunda alternativa o Music Service faz de *relay server* e redireciona a *stream* para o respetivo cliente.

Durante o *streaming* de áudio não é efetuado apenas um envio dos *chunks*, uma vez que, o utilizador pode controlar a reprodução decidindo, por exemplo, pausar a música. Ao efetuar esta ação não deve ser apenas colocada em pausa a reprodução musical, mas também o envio de novos *chunks*, a partir do momento em que o *buffer* tenha atingido a sua capacidade máxima.

Devido à necessidade de controlar o fluxo de dados durante a reprodução musical são estabelecidas inicialmente duas ligações, uma para receber comandos de controlo, numa porta pré-definida e já do conhecimento do Client App, ao passo que a outra apenas envia

os *chunks* referentes ao ficheiro musical, aberta numa porta aleatória cuja informação será transmitida posteriormente para o cliente. Uma estratégia semelhante durante a transferência de ficheiros por parte do protocolo *FTP*.

De salientar que, quer o *streaming* seja efetuado diretamente, ligação estabelecida entre Client App e Local Agent, quer indiretamente, Music Service como ponto de ligação, os princípios são os mesmos, pelo que as mensagens de controlo e a forma como o conteúdo musical é processado é exatamente o mesmo, já que apenas os pontos terminais da ligação é que interpretam as mensagens e processam as músicas. A única diferença reside na forma como são enviados os dados, já que indiretamente primeiro tem de passar pelo Music Service antes de chegar ao destino. Desta forma, a estratégia a seguir descrita sobre como o processo de *streaming* foi implementado é comum aos dois tipos de *streaming*.

4.7.1 *Validação da Sessão de Streaming*

Quando um Client App estabelece uma conexão com o servidor de *streaming*, Local Agent no caso de ser uma *stream* direta e o Music Service no caso de uma *stream* indireta, é enviado um pedido para o Music Service para se autenticar na *stream*.

A autenticação da *stream* é um processo que visa obter dados que permitam validar o acesso de conteúdo de um cliente durante a transmissão dos dados. Neste pedido, efetuado por HTTPs, o Music Serive valida se o Client App tem acesso ao conteúdo do Local Agent, e caso seja validado, envia como resposta os seguintes dados:

SESSION ID: um identificador único para a sessão de *streaming* que envolve um Client App e um Local Agent;

SESSION PASSWORD: a palavra-chave da sessão (ou segredo da sessão).

Após a autenticação na sessão de *streaming*, a troca de mensagens durante a transferência dos dados é realizada por uma comunicação, via TCP, que não está cifrada. Esta decisão justificasse porque o conteúdo trocado não é sensível, logo não faria sentido acrescentar uma camada de proteção de dados que acrescentaria um *overhead* de segurança afetando a performance no fluxo de dados.

No entanto, juntamente com o conteúdo da mensagem é enviado o identificador único da sessão, Session ID, e uma *hash* obtida através da palavra-chave da sessão e da informação a ser enviada. Independentemente da origem da mensagem, durante uma sessão de *streaming*, estes dados devem ser sempre incluídos para que se possa efetuar a validação da sessão.

Aquando da receção das mensagens é efetuada uma validação dos dados recebidos com a extração da palavra-chave de sessão associada ao identificador único disponibilizado e ainda os intervenientes da mesma.

No caso do Client App, após a autenticação na *streaming*, este armazena os dados de sessão localmente, só consultando o Music Service novamente se os mesmos expirarem ou se iniciar uma nova sessão com um Local Agent diferente.

Quando o Local Agent recebe uma mensagem de *streaming* cuja informação sobre a sessão não esteja presente localmente, efetua um pedido ao Music Service, via HTTPs, para obter os dados da sessão (palavra-passe). Após a validação da sessão com a formação da *hash*, através da palavra-passe de sessão e do conteúdo da mensagem, os dados são guardados localmente e é enviada a resposta para o cliente. A partir deste momento como os dados da sessão são guardados localmente não é necessário realizar um pedido ao Music Service para os obter, a não ser que a sessão expire.

De notar que, no caso do Local Agent a informação que obtém sobre a sessão é apenas a palavra-passe associada à mesma, uma vez que no Music Service é efetuada a validação de que o Local Agent que requisita o pedido está incluído na sessão indicada e que o Client App que inicia a sessão de *streaming* pode obter os dados do Local Agent. Desta forma, o Local Agent apenas tem de enviar a informação requisitada, sem ser necessário este saber a qual o cliente deve enviar.

No caso do Music Service é efetuada uma validação semelhante à do Client App e Local Agent aquando da recepção de uma mensagem de *streaming*, com a comparação da *hash* fornecida no corpo da mensagem e a *hash* formada localmente. Porém, não necessita de efetuar o pedido a nenhum componente uma vez que a informação está armazenada localmente.

No caso de alguma mensagem não ser validada esta será descartada, uma vez que a informação contida na mesma não é válida. Este mecanismo de validação permite aumentar a segurança durante a sessão de *streaming* utilizando uma via de comunicação que não está encriptada. No entanto, não protege contra ataques *Man In The Middle*, no entanto qualquer alteração das mensagens é detetada e a mesma descartada, o que, por exemplo, faz com que não sejam respondidos a pedidos que possam ter sido alterados por terceiros.

4.7.2 *Processamento do Conteúdo Musical*

Do lado do cliente a música é reproduzida utilizando a biblioteca AudioTrack [62]. Esta reproduz conteúdo que está armazenado num *buffer*, permitindo que se escreva no mesmo, através de uma *API*, sem que esta ação interrompa a leitura e consequente reprodução do conteúdo. Porém, esta biblioteca apenas é compatível com dados *PCM*, ou seja, conteúdo musical no formato WAV.

Para reproduzir ficheiros no formato WAV, utilizando a biblioteca referida acima, não é necessário efetuar qualquer ação, mas ficheiros dos formatos MP3 ou FLAC precisam de ser convertidos para WAV antes da sua reprodução. Esta conversão pode ocorrer tanto do lado

do cliente como do lado do servidor. Uma vez que as bibliotecas utilizadas para converter ficheiros do formato MP3 e FLAC para WAV, JLayer (para MP3) [63] e nayuki (para FLAC) [64], têm como *input* o ficheiro na sua totalidade, implica que este trabalho seja efetuado do lado do servidor, já que apenas este tem acesso total ao conteúdo musical. Do lado do cliente, como recebe apenas pedaços correspondente ao ficheiro musical, torna inviável a conversão aquando da receção do conteúdo, sendo possível reproduzir diretamente o conteúdo recebido já que o mesmo já vem processado.

Uma desvantagem desta abordagem é que é transferida uma maior quantidade de dados já que ficheiros com dados PCM têm um armazenamento sem dados comprimidos, ou seja, sem perdas. Recordando um exemplo anterior, 10 segundos de música em *MP3*, com um *bitrate* de 128kbps, equivalem a 160KB, ao passo que o mesmo intervalo de tempo num formato WAV, a uma frequência de 44.1KHz (qualidade de CD), é de aproximadamente 2MB. Para contornar esta abordagem foi necessário efetuar a compressão dos dados a ser enviados. Como a música é repartida em diferentes *chunks* faz com que a compressão seja aplicada a dados que ocupam um espaço de armazenamento mais reduzido quando comparado com o ficheiro na totalidade (convertido para WAV) pelo que o processo de compressão ou descompressão não afeta significativamente a latência na reprodução.

Pelo facto de a conversão ser efetuada do lado do servidor faz com que do lado do cliente quando recebe o conteúdo, e após a sua descompressão, o mesmo se encontra pronto para ser escrito diretamente no *buffer*. De salientar, que antes da reprodução da música é efetuada uma configuração do *player* musical de acordo com a meta informação disponibilizada pelo servidor. Esta configuração pode alterar de música para música, pelo que no final da reprodução esta configuração é eliminada, bem como os recursos do AudioTrack.

A leitura do conteúdo musical é efetuada tendo em conta uma medida temporal, uma vez que do lado do cliente, por uma questão de comodidade, o *buffer* é expresso em função do tempo (neste caso em segundos) ao invés do seu espaço de armazenamento. Desta forma, o ficheiro, do lado do servidor, é lido *frame* por *frame* tendo sempre a informação de qual intervalo temporal a que o *frame* pertence.

Uma vez que apenas do lado do cliente é que tem acesso ao *buffer* temporal, é este que controla a reprodução do conteúdo. Quando a capacidade do *buffer* baixa dos 80% este requisita um novo espaço temporal ao servidor (Local Agent).

Do lado do cliente, como já foi referido anteriormente, a reprodução é efetuada com o auxílio da biblioteca AudioTrack que implementa um *buffer*. Porém, na sua vasta *API* não disponibiliza quaisquer métodos para obter informação do mesmo, pelo que não há uma forma precisa de saber quanto espaço resta no *buffer*. Desta forma, foi necessário efetuar um controlo aproximado do mesmo, uma vez que é possível saber quanto tempo foi escrito no *buffer* do AudioTrack e quanto tempo já passou desde o início da reprodução musical. Apesar de esta medida não ser a ideal, permite realizar um controlo com precisão ao nível do

segundo para controlar o *streaming*, que é suficiente para a maioria da duração dos áudios a serem reproduzidos.

4.7.3 *Mensagens de Controlo*

O utilizador tem a capacidade de controlar a reprodução musical. Para tal, dispõe de uma série de ações:

PLAY: inicia a reprodução da música requisitada, se alguma sessão de *streaming* estiver ativa no dispositivo é imediatamente terminada;

PAUSA: pára a reprodução musical, no entanto o *streaming* continua ativo;

RETOMAR: ação apenas executada se o *streaming* estiver no estado de pausa, para continuar a reproduzir o conteúdo no momento temporal em que foi pausado;

AVANÇAR: é alterado o tempo de reprodução musical, para um momento posterior ao atual;

RECUAR: é alterado o tempo de reprodução musical, para um momento anterior ao atual;

PARAR: pára a reprodução musical e desativa o *streaming*.

Por cada ação tomada por parte do utilizador, não só é despoletado um evento para alterar o estado do *player*, como também é enviada uma mensagem para o Local Agent para alterar o estado do *streaming*. Foram definidas um conjunto de mensagens correspondentes às ações descritas anteriormente.

Request Music

Enviada pelo cliente quando o utilizador escolhe uma música para reproduzir. Estrutura da mensagem:

NAME: nome da música requisitada;

FIRST INTERVAL: referente ao início da reprodução musical, i.e., o momento temporal em que a transferência deve iniciar. Por defeito, o valor é zero em concordância com o início de uma nova música, no entanto, nos casos em que uma sessão de *streaming* seja mantida do lado do cliente, pode ser requisitado o início de uma música com um tempo de reprodução diferente do início. A medida do tempo é em milissegundos.

Quando o Local Agent recebe uma mensagem deste tipo e caso não tenha estabelecido ainda uma ligação para enviar os dados para este cliente, abre uma conexão TCP numa porta aleatória, entre 51000 e 51100, e quando encontra uma disponível, envia uma mensagem

de StartStreaming com informação do número da porta ao qual o cliente se deve conectar. Para além disto, inicia a leitura e envio do ficheiro desde o intervalo requisitado. Se por alguma razão não for possível iniciar o *streaming*, por exemplo, a música requisitada não existe, é despoletado um erro e enviada uma mensagem StreamInvalid com a razão pelo qual o servidor não consegue iniciar a transferência do ficheiro.

No Client App após o envio de uma mensagem de RequestMusic espera pela resposta do servidor. Caso seja negativa, apresenta o erro ao utilizador ou toma as medidas necessárias para o contornar, dependendo do tipo do mesmo. No caso afirmativo, configura o *player* consoante as caraterísticas da música, por exemplo o *sample rate* do som, e estabelece a ligação na porta indicada pelo Local Agent para receber os *chunks* referentes ao conteúdo musical.

Request Music Interval

Enviado pelo cliente para o servidor, sempre que o utilizador alterar o tempo de reprodução de uma música. Este tipo de mensagem é utilizado, tanto para avançar no tempo como recuar, uma vez que os princípios das duas ações são os mesmos. A estrutura da mensagem é a seguinte:

START INTERVAL: referente ao novo espaço temporal que a transferência deve retomar em relação ao tempo total da música e não o momento atual do servidor, já que pode ser diferente do cliente. Esta medida é expressa em milissegundos;

LENGTH: quantidade de tempo a ser enviado em milissegundos.

No momento em que o Local Agent recebe uma mensagem de controlo deste tipo pausa imediatamente a *stream* e avança na leitura do ficheiro até ao momento que foi requisitado. Nos casos, em que o tempo requisitado é anterior ao atual lido do ficheiro, então é iniciada uma nova leitura até chegar ao tempo pedido. Não é possível avançar imediatamente para um intervalo de tempo porque a leitura do ficheiro é feita *frame* por *frame* o que torna a sua leitura sequencial. Se for requisitado um intervalo de tempo distante do atual pode impor uma maior latência durante a transmissão, mas sem afetar a experiência de utilização por parte do servidor.

Quando o utilizador altera o momento atual de reprodução de uma música, no Client App é alterado as variáveis de controlo do *buffer* para atualizar para o momento atual e eliminar os espaços temporais que estão armazenados no *buffer* dado que os mesmos, em princípio, não serão reproduzidos por não serem consequentes ao novo intervalo temporal requisitado.

É ainda necessário descartar os *chunks* do ficheiro que ainda podem estar em trânsito ou armazenados no *buffer* do TCP que não correspondem ao novo intervalo requisitado. Esta

ação é possível ser tomada, uma vez que no controlo do *buffer* possui informação sobre a quantidade temporal que foi requisitada e ainda não foi reproduzida.

Pause Streaming

Quando o utilizador executa a ação de pausa na reprodução, é enviada esta mensagem para o Local Agent com o corpo vazio, uma vez que o próprio tipo da mensagem é suficiente para indicar a ação a ser tomada.

No lado do servidor não são mais enviados novos segmentos de música até que seja dada uma ordem em contrário por parte do cliente. No Client App é colocado em pausa o *player* AudioTrack e o controlo do tempo de transmissão musical, uma vez que o tempo de reprodução está parado.

Stop Streaming

À semelhança da mensagem de Pause Streaming, quando o utilizador despoleta um evento para parar a *streaming* é enviada uma mensagem de Stop Streaming com o corpo vazio.

Tanto do lado do cliente como do servidor, são eliminados os dados referentes à sessão de *streaming* e é fechada a conexão estabelecida inicialmente para enviar os dados. De notar que, a ligação para enviar os comandos de controlo continua aberta dado que pode ser requisitado o início de uma nova música.

Next Segment

Este tipo de mensagem é enviado automaticamente para o servidor sempre que o *buffer* tem um espaço livre, neste caso, sempre que a sua capacidade temporal baixa dos 80% da capacidade total. Desta forma, garante-se que o *buffer* tem conteúdo suficiente para continuar a reproduzir o conteúdo caso a ligação seja interrompida momentaneamente. A estrutura da mensagem é constituída apenas por um campo:

LENGTH: quantidade de tempo a ser enviado em milissegundos.

Como já foi indicado anteriormente o Client App constrói e envia este tipo de mensagem sempre que o *buffer* relativo ao tempo musical se encontra a menos de 80% da sua capacidade. No lado do servidor, Local Agent, sempre que recebe esta mensagem lê o próximo segmento temporal a partir do último lido. Caso a *stream* esteja no estado de pausa, é automaticamente retirada desse estado e colocada de novo em ativa.

Quando a leitura do ficheiro chega ao fim, quer através de uma mensagem de Next Segment ou então com uma Request Music Interval, a *stream* é imediatamente terminada já que não serão transferidos mais *chunks*.

Request Stream e Control Message Wrapper

Como na utilização da tecnologia Google Protobuffers em Java não é possível validar o tipo de uma mensagem através da extração direta dos *bytes* foi necessário encapsular as mesmas noutra mensagem, que para além da mensagem com os dados continha que tipo de mensagem era para que fosse possível efetuar a extração e identificação da mesma. A mensagem que encapsula todas as mensagens de controlo designa-se de Control Message Wrapper.

Como foi mencionado anteriormente, associado aos dados da mensagem está presente a informação sobre a sessão de *streaming*, com o identificador único da mesma e uma *hash*, formada pelo conteúdo e a palavra-chave de sessão. Neste sentido, é enviada uma mensagem de Request Stream:

SESSION ID: identificador único da sessão de *streaming*;

HASH: formada pela *password* de sessão e o conteúdo da mensagem, para efetuar a validação dos dados no destino;

CONTROL MESSAGE WRAPPER: conteúdo da mensagem, com os comandos de controlo.

4.7.4 *Streaming Indireto*

O *streaming* indireto é um recurso para a transferência do conteúdo, ou seja, só é executado se não for possível efetuar o *streaming* com a estratégia definida anteriormente. Tal pode acontecer quando não é possível estabelecer uma ligação direta entre o Local Agent e o Client App, por exemplo, devido a bloqueios das respetivas *firewalls*. Neste processo estão, sempre, envolvidas três componentes:

- O Local Agent que é responsável por disponibilizar o conteúdo;
- O Client App que vai consumir o conteúdo transferido;
- O Music Service cujo papel é redirecionar o conteúdo enviado do Local Agent para o respetivo Client App.

A diferença significativa entre o *streaming* indireto e o *streaming* por defeito, que pode ser apelidado de *streaming* direto, é que o Music Service apresenta um papel fundamental para estabelecer uma ligação entre o produtor e consumidor de conteúdos.

Como, tanto o Local Agent como o Client App, são dispositivos pessoais, sobre o qual o sistema desenvolvido não tem qualquer controlo, podem existir restrições nas ligações que estabelecem que podem impedir a ligação com outros sistemas. Porém, qualquer dispositivo

que se conecte ao sistema tem de se conectar ao Music Service pelo que estes conseguem se conectar a este componente, não existindo o problema de bloqueio da ligação.

No entanto, um dispositivo pode enfrentar problemas de rede ou então, simplesmente, está desconectado, o que impede o estabelecimento de uma ligação com o Music Service. Mas neste caso em concreto, o dispositivo não consegue comunicar e é dado como inativo, pelo que não é possível arranjar uma solução para efetuar o *streaming* nestes casos.

O Music Service vai ter o papel de um servidor de *relay*, uma vez que será a ponte de ligação entre os dois componentes e que reencaminha os pacotes para o destino correto. No entanto, o protocolo de *streaming* não sofre alterações significativas aquando da realização da transferência do conteúdo utilizando esta metodologia.

É mantida a ideia da existência de dois canais de comunicação, um para envio das mensagens de controlo e outro para envio dos dados. No entanto, no Music Service são estabelecidas quatro conexões, tal como mostra na figura 12.

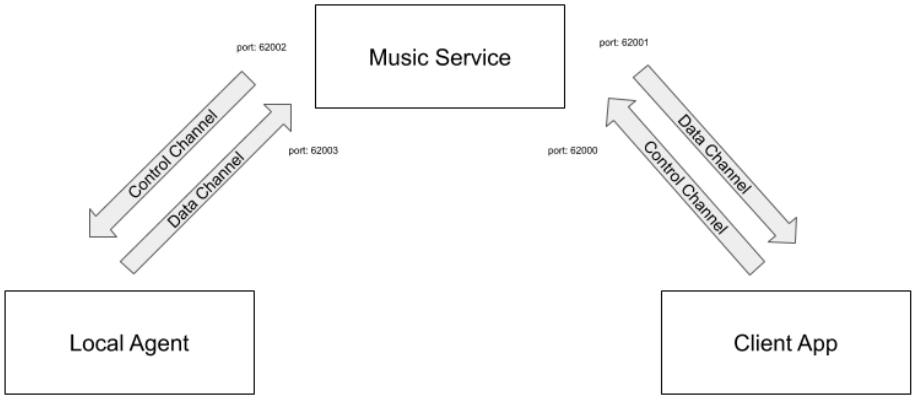


Figura 12: *Streaming* Indireta - Canais de Ligação

Da figura 12 percebesse que o Music Service tem o papel de servidor, uma vez que é o mesmo que possui portas pré-definidas e os dispositivos é que devem de estabelecer a conexão a este componente. De notar que, é efetuada uma separação entre o Local Agent e o Client App para identificar, mais facilmente, que tipo de dispositivo é que se está a conectar ao servidor principal, Music Service, para a realização da transferência de conteúdo.

Neste processo, o Client App tem na mesma um papel de Client, sendo que intuitivamente o Music Service funciona como um servidor para este componente. No entanto, quem providencia o conteúdo é o Local Agent e quem lhe solicita o conteúdo é o Music Service,

pelo que a relação entre estes dois componentes é de, respetivamente, Servidor-Cliente. No entanto, como o Music Service é o componente principal de todo o sistema, não sendo controlado por terceiros, faz sentido que seja o Local Agent a estabelecer uma conexão ao Music Service apesar de ser este último a solicitar o conteúdo.

No Music Service é guardado, para cada sessão de *streaming*, os canais de comunicação envolvidos. Sendo que, como é possível perceber da figura 12, o canal de controlo do Client App e o canal de dados do Local Agent são canais apenas de leitura, ao passo que, o canal de dados do Client App e o canal de controlo do Local Agent são canais em que o Music Service apenas escreve.

Como já foi referido, o protocolo para a realização do *streaming* mantém-se inalterado, no entanto foram acrescentadas novas mensagens de controlo que serão só interpretadas pelo Music Service para efetuar o redireccionamento dos pacotes para os destinos corretos e identificação da origem dos canais de comunicação.

Music Service Start Streaming

Mensagem enviada do Music Service para o Local Agent numa primeira solicitação de ligação de um cliente e permite informar o Local Agent que vai ser realizado um *streaming* indireto e que este se deve de conectar ao Music Service. A mensagem é constituída por:

CONTROL PORT: porta do canal de mensagens de controlo à qual o Local Agent se deve de ligar;

DATA PORT: porta do canal dos dados à qual o Local Agent se conecta.

Este tipo de mensagem é efetuado apenas uma única vez, aquando do estabelecimento de ligação de um novo cliente com o Music Service. Tem como objetivo informar o Local Agent que vai ser iniciado um novo *streaming* e que, ao contrário de uma transferência de dados efetuada diretamente para o Client App, O Local Agent deve-se conectar ao Music Service para efetuar a transferência do conteúdo.

Connected

Mensagem especial que permite identificar o estabelecimento de uma ligação, tanto de um Local Agent como de um Client App. Esta mensagem é enviada sempre que um Client App se conecta no canal de dados ao Music Service e quando um Local Agent se conecta ao Music Service (tanto no canal de dados como no canal das mensagens de controlo).

Esta mensagem é necessária porque ao contrário da conexão que o Client App estabelece, no canal de controlo de dados, utilizando a mensagem protocolo Request Stream que o permite identificar de imediato, nos restantes canais se nada fosse feito, os dispositivos

estabeleciam uma conexão e seria complexo para o Music Service conseguir descobrir a identificação do dispositivo.

Desta forma, após a conexão dos dispositivos, ou seja, o Local Agent e o Cliente App nos respetivos canais, enviam esta mensagem protocolar para se identificarem e para que o Music Service possa guardar os canais associados à sessão de *stream* correta. A mensagem Connected é constituída por:

SESSION ID: identificador único da sessão de *streaming*;

HASH: formada pela *password* de sessão e o conteúdo da mensagem, para efetuar a validação dos dados no destino.

Transferência de Dados

A implementação da transferência dos dados no *stream* indireto não foi alterada, tanto no Local Agent como no Client App. No entanto, o Local Agent envia os dados, numa primeira instância para o Music Service e este tem a responsabilidade de redirecionar para o Client App correspondente, como seria de esperar.

O redireccionamento dos dados referentes ao conteúdo é conseguido porque quando o Local Agent estabelece a conexão, no canal de dados, com o Music Service envia uma mensagem protocolar Connected que contêm a sua identificação e a do Client App. A partir deste ponto, o Music Service extrai a informação sobre a sessão de *streaming* que necessita, nomeadamente a conexão, no canal de dados, estabelecida com o Client.

Esta ligação estabelecida com o Local Agent vai ser única e exclusivamente utilizada para enviar dados para o Client App identificado inicialmente. Se um outro cliente diferente pretender efetuar um *streaming* com o mesmo Local Agent, através do Music Service, são sempre estabelecidas novas ligações entre estes dois componentes, daí se assegurar a exclusividade de cada conexão para apenas um Client App.

4.8 ARMAZENAMENTO DE INFORMAÇÃO NO MUSIC SERVICE

O sistema de armazenamento do Music Service é responsável por armazenar a meta-informação do conteúdo que cada servidor privado (Local Agent) disponibiliza para os diferentes utilizadores.

Numa fase inicial do sistema todo o conteúdo de um Local Agent era guardado na base de dados associado ao mesmo. No entanto, como está a ser considerado conteúdo pessoal de diferentes utilizadores pode perfeitamente ocorrer a situação em que dois Local Agents diferentes possuam o mesmo conteúdo.

Apesar de ser uma abordagem bastante simples não é possível de utilizar uma vez que não existiria espaço disponível, com uma replicação desmedida dos dados. Se todos os Local

Agents possuírem informação, por exemplo, sobre um mesmo álbum, esta iria aparecer N vezes na base de dados (sendo que N é o número de Local Agents registados no sistema).

Desta forma, foi necessário implementar uma estratégia que visa a eliminação de duplicação de informação no sistema. De notar que a meta-informação obtida pelo Music Service de um determinado Local Agent é da responsabilidade do respetivo dono do servidor privado, o que leva a que um mesmo elemento musical possa entre diferentes servidores possa ter representações diferentes. Por exemplo, num servidor privado pode existir o gênero musical *rock* e noutro *rock & roll*, apesar das diferenças nos nomes, para um ser humano é evidente que estas duas designações se referem ao mesmo gênero.

Os dados do conteúdo a disponibilizar são referentes a meta-informação musical que é caraterizada pelos seguintes elementos:

GÊNERO: carateriza o estilo musical;

ARTISTA: autor musical que está associado a vários gêneros;

ÁLBUM: coleção de músicas que estão relacionadas a um ou mais artistas;

PARTIÇÃO DO ÁLBUM: divisão do álbum por partes que inclui músicas relacionadas (nem todos os álbuns possuem partições);

MÚSICA: conteúdo musical que é representado por todos os elementos anteriores, para além das suas caraterísticas (como por exemplo o nome).

A inserção de informação na base de dados do Music Service é realizada em três fases:

1. **Extração e normalização dos dados**: leitura do *filesystem* da meta-informação do conteúdo, com respetiva normalização (por exemplo: converter todas as letras para minúsculas). Fase a ser realizada no Local Agent;
2. **Agregação e Envio da Informação**: envio de um conjunto de dados a partir do Local Agent para o Music Service, via HTTPs;
3. **Inserção dos dados**: identificação de duplicados no sistema de armazenamento e inserção ou atualização da informação, no Music Service.

4.8.1 *1ª Fase - Extração e Normalização dos Dados*

O primeiro passo de todos é a leitura no *filesystem* no caminho especificado pelo utilizador. Esta estratégia já foi detalhada na secção 4.3.

A definição dos nomes está ao cargo do utilizador que possui o conteúdo. Diferentes pessoas fazem com que existam diferentes abordagens nas designações que caraterizam os

elementos. Por exemplo, ao passo que um utilizador decide usar *Jazz Rock* um outro usa *jazz_rock*. Ambos caraterizam um mesmo gênero musical, mas num há o uso de maiúsculas e minúsculas ao passo que outro são só usadas letras minúsculas e no primeiro a divisão das palavras é efetuada através de espaços enquanto no segundo é utilizado o caractere especial "_".

Neste sentido é necessário realizar uma normalização dos nomes para que estes sejam uniformes e assim atingir melhores resultados no processo de obtenção do grau de similaridade entre os diferentes textos. A normalização consiste na execução dos seguintes passos:

- Conversão de dígitos para a sua extensão por escrito. Exemplo: 420 seria convertido em *four hundred and twenty* (foi utilizada a extensão em inglês por ser uma língua universal);
- Remoção de acentos e caracteres especiais (que serão substituídos por espaços). No caso do caractere "&"é substituído pela palavra *and*;
- Remoção dos espaços presentes no início e fim das palavras, bem como a concatenação de múltiplos espaços consecutivos entre palavras em apenas um só espaçamento;
- Conversão de todas as maiúsculas para minúsculas;
- Conversão dos caracteres *unicode* para os caracteres mais semelhantes em UFT-8.

Para realizar a normalização dos nomes a linguagem Java já possui várias bibliotecas embutidas que permitem utilizar expressões regulares e desta forma efetuar a conversão desejada de uma forma simples.

4.8.2 *2ª Fase - Agregação e Envio*

A meta-informação extraída sobre o conteúdo é guardada localmente em cada servidor privado e, só depois de terminado este processo, é que são enviados os dados para o Music Service. De notar que, o Local Agent vai processar a informação de forma a eliminar duplicados localmente, a estratégia utilizada é a mesma que será utilizada no Music Service e que será explicada com detalhe adiante.

Após a conclusão da 1ª fase, a informação é agregada por artista e álbum e enviada para o Music Service, sendo que cada mensagem é constituída pelos seguintes campos:

- lista dos gêneros musicais (constituída pelo nome dos elementos);
- nome do artista;
- nome do álbum;

- lista de outros álbuns pertencentes ao artista (constituída pelo nome dos elementos);
- lista de caraterísticas, nome, duração e partição de álbum (se existir), das músicas pertencentes ao álbum.

Como cada entidade musical pode ter associado vários nomes, de forma a reduzir a informação disponibilizada, para o Music Service é enviado apenas o nome mais frequente para o respetivo elemento musical. O mesmo acontece com as músicas onde é enviado o conjunto de caraterísticas (nome, duração e partição de álbum) que aparece mais vezes na base de dados. Em caso de empate aleatoriamente é escolhida uma caraterística.

Por cada artista e álbum é criada uma mensagem que é enviada para o Music Service via HTTP.

4.8.3 *3ª Fase - Procura e Inserção ou Atualização*

Na terceira fase realizasse a inserção de novos dados ou atualização dos existentes no Music Service. Esta tomada de decisão vai se basear num algoritmo que calcula a similaridade do conteúdo a ser inserido com os dados existentes no sistema. Neste algoritmo, o principal objetivo é verificar se as canções a inserir já existem, e todos os seus elementos, existem na base de dados.

Simplificadamente o algoritmo divide-se em duas partes. Numa primeira parte todos os dados que constituem uma canção entram na equação para verificar se a mesma existe, neste caso, o artista, álbum, partição do álbum e as caraterísticas ligadas diretamente à música. O gênero não é considerado devido porque na maioria dos casos não é bem definido por parte dos utilizadores.

Numa segunda parte é efetuada uma avaliação individual dos constituintes da música, quer que se tenha confirmado que a música existe ou não na base de dados.

De modo a explicar o funcionamento geral do algoritmo, está ilustrado na figura 13 uma representação em grafo dos dados armazenados no sistema que vai ser usada para mostrar um exemplo prático do processo de inserção.

O novo conteúdo que vai ser alvo de análise é constituído pelos seguintes elementos musicais:

- um artista com a designação de *Pink Floyid*;
- um álbum com a designação de *Dark side of the moon*;
- uma música com o nome de *Shine On You Crazy Diamond, Welcome To The Machine* e um tempo de duração de 21 minutos e 3 segundos.

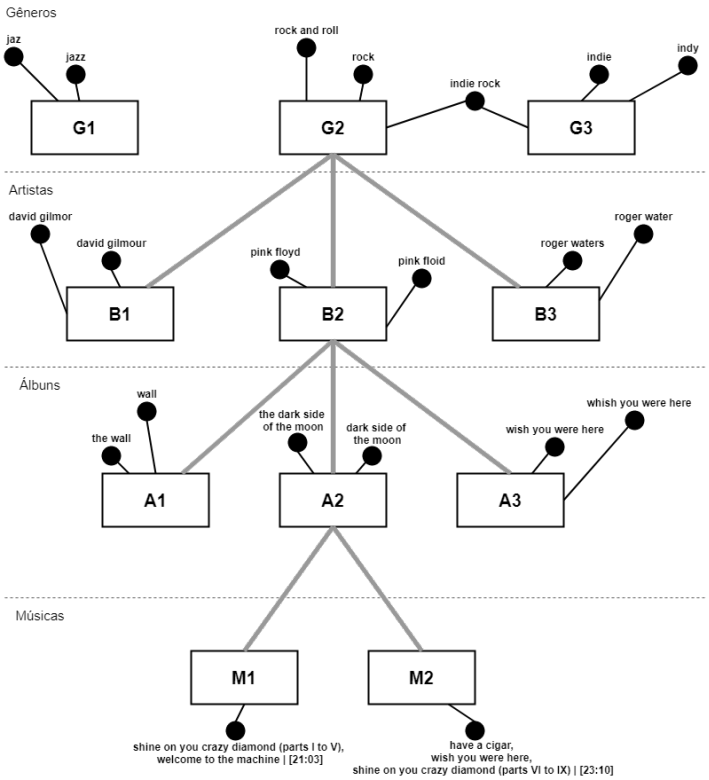


Figura 13: Representação de Informação Armazenada no Sistema

O que o algoritmo vai calcular é a similaridade do artista e do álbum (a música está intrinsecamente ligada ao álbum) a inserir com os artistas e respetivos álbuns presentes na base de dados do sistema. Imagine-se que, após a execução do algoritmo, se obtém um valor de similaridade de 90% para a canção que se quer inserir, que está relacionada com a seguinte (extraída do grafo):

- artista B2
- álbum A2
- música M1

Apesar de o valor de similaridade não ser 100%, que nos indicava claramente que existia uma cópia igual na base de dados, é um valor relativamente próximo de 100, o que nos indica que é muito provável que o par artista mais álbum que se pretende inserir já existem na base de dados.

No exemplo apresentado não foi obtido 100% de similaridade pelo facto de não existir uma total correspondência entre os nomes dos artistas e os nomes das músicas serem diferentes. No entanto a informação é tão semelhante que é muito provável que sejam os mesmos elementos.

Após ser encontrado os elementos musicais do sistema mais semelhantes aos que se pretende inserir, é necessário verificar se no sistema os elementos já possuem as caraterísticas iguais aos novos elementos. Por exemplo, no caso do artista com o identificador B2 iria ser associado um novo nome, *Pink Floyid*.

Imagine-se outro cenário em que o conteúdo que se pretende introduzir é o seguinte:

- um artista com o nome de *Pink Floyd*;
- um álbum com o nome de *Animals*;
- duas músicas, *Pigs On The Wing (Part I)* e *Dogs*, com um tempo de duração respetivamente de 1 minuto e 26 segundos e 17 minutos e 5 segundos;
- adicionalmente existe a informação sobre mais três álbuns associados ao artista: *The wall*, *Dark side of the moon* e *Wish you were here*.

Neste exemplo existe mais informação sobre o artista, uma vez que são passados mais dados sobre os álbuns que o mesmo possui. No entanto, as músicas estão associadas apenas a um álbum (esta informação é sempre representada no pedido que se efetua, como foi demonstrado anteriormente).

Imagine-se que após a execução do algoritmo, o maior valor de similaridade obtido foi de 40%. Ora, este valor é bastante baixo, o que leva a inferir que ainda não existe aquele par artista mais álbum no sistema.

No entanto, neste cenário não se pode concluir que vai ter de ser acrescentada toda a informação, ou seja, um novo artista, um novo álbum e novas músicas. Desta forma, no algoritmo é efetuada uma avaliação individual dos elementos musicais.

Com esta avaliação individual infere-se que há um artista, com o identificador B2, que é muito similar ao artista que vem no pedido. A única informação nova que vai ser acrescentada é o álbum, com por exemplo o identificador A4 associado a um novo nome *Animals* e duas novas músicas com, por exemplo, o identificador M3 e M4.

De seguida, serão apresentados os detalhes sobre quais são os cálculos e a informação que estão envolvidos na obtenção de um valor de similaridade, bem como qual o valor de similaridade mínimo ótimo para verificar se uma canção ou um elemento já existem no sistema.

3ª Fase, 1ª Parte - Procura de Duplicados

A informação enviada pelo Local Agent já se encontra agrupada por artista e álbum, com as caraterísticas de ambos, as caraterísticas das músicas associadas ao álbum e ainda os gêneros. Desta forma, o que vai ser executado é uma procura nos dados do Music Service no sentido de encontrar as canções com uma ênfase no artista e no álbum, uma vez que são os elementos com mais informações.

A meta-informação a armazenar é baseada em caraterísticas que foram previamente definidas pelo dono da coleção. Estas são maioritariamente nomes. No entanto um nome não é suficiente para identificar um elemento, como foi provado nos exemplos anteriores. Assim, para avaliar a existência de informação de duplicados no sistema foram definidas as seguintes regras:

- No caso do gênero como é uma entidade muita das vezes mal atribuída e que se diferencia muito de utilizador para utilizador, este não será considerado na equação de procura de duplicados das canções. Porém, é sempre feita uma avaliação se um dado gênero já se encontra no sistema;
- No caso do artista é utilizado o seu nome juntamente com a informação dos álbuns, para validar a sua existência no sistema;
- O mesmo se processa com o álbum que para além do seu nome é considerado as caraterísticas das músicas que lhe estão subjacentes;
- Para cada música é efetuada uma comparação com as caraterísticas que lhe estão associadas diretamente, como o seu nome e o tempo de duração.

Pelo que foi indicado anteriormente, ao efetuar a procura por um conjunto de canções há uma maior ênfase em encontrar o artista e o álbum, como um par. Porém, como na maioria dos casos se tem informação mais detalhada sobre o álbum então este elemento é que possui um maior peso na equação. Assim a fórmula do cálculo da existência de informação referente a um conjunto de dados disponibilizados pelo Local Agent, é o seguinte:

$$\text{similaridade\_elementos} = 0.35 * \text{similaridade\_artista} + 0.65 * \text{similaridade\_album} \quad (1)$$

Como os valores da similaridade do artista e do álbum estão situados num domínio fechado entre 0 e 1, então a similaridade dos elementos também se encontra. Quanto mais perto a similaridade estiver de 1 significa que é mais semelhante.

Similaridade dos Nomes

Todos os elementos musicais possuem caraterísticas, diretamente relacionadas com os mesmos. No caso de gênero, artista e álbum apenas está a ser armazenado os nomes associados aos elementos, pelo que é a única caraterística a ser considerada. No entanto, no caso da música para além de ser caraterizada pelo nome também o é pelo seu tempo de duração.

Como a definição dos nomes está a cargo de terceiros ao sistema, em concreto, é da responsabilidade dos donos dos servidores privados, existe a possibilidade da ocorrência de

erros ou que então dois nomes que sejam diferentes, caractere a caractere, sejam na verdade muito semelhantes e que possam estar a representar a mesma informação.

Um exemplo prático desta situação acontece quando um utilizador escolhe *rock & roll* para caraterizar um gênero musical, ao passo que outro utiliza *rock and roll*. Se fosse aplicado um algoritmo tradicional de igualdade de textos para comparar os nomes, então teríamos como resultado que os mesmos são diferentes e cada um deles estaria a representar um gênero musical diferente. No entanto, os dois textos são semelhantes e até representam o mesmo gênero musical, logo não deveriam estar a representar gêneros musicais diferentes no sistema, mas sim estarem ambos associados a um mesmo gênero musical.

Por esta razão, é utilizado um algoritmo de distâncias de textos, que determina o grau de similaridade entre dois textos fornecidos, para calcular a similaridade entre nomes. A estratégia que foi implementada para encontrar o grau de similaridade denomina-se de *fuzzy search* e é utilizada nos motores de pesquisa para efetuar as recomendações de palavras quando é inserida alguma gramaticalmente incorreta ou para efetuar associações entre palavras que sejam semelhantes.

Cálculo de Similaridade do Artista e do Álbum

O cálculo de similaridade do artista é obtido através das caraterísticas que lhe estão diretamente ligadas, ou seja, o seu nome. No entanto, não se pode considerar apenas o nome do artista para verificar a existência deste, já que, anteriormente foi demonstrado que só o nome não é suficiente para identificar unicamente um elemento musical.

O cálculo do artista envolve não só os nomes associados ao mesmo como também o valor de similaridade dos álbuns que lhe estão associados. Sendo que o valor de similaridade é obtido através da seguinte fórmula:

$$\text{similaridade\_artista} = 0.5 * \text{similaridade\_do\_nome} + 0.5 * \text{similaridade\_nome\_dos\_albuns} \tag{2}$$

Como é possível observar pela fórmula anterior é necessário efetuar dois cálculos para encontrar a similaridade para cada um dos artistas presentes no Music Service. Assim, o que permite identificar um artista entre componentes no sistema são os nomes que lhe estão associados juntamente com os álbuns que o mesmo possui, e não apenas um nome.

Inicialmente é efetuada uma extração, na base de dados do Music Service, de todos os artistas ficando com um conjunto formado pelos identificadores únicos e para cada, um conjunto de nomes que lhe estão associados bem como os seus álbuns.

Para cada elemento é executado o algoritmo de similaridade de textos sobre os seus nomes, comparando-os com o nome do artista que se pretende inserir. De seguida, é selecionado o nome com o maior valor de similaridade. Desta forma obtém-se a primeira parte da equação

para o cálculo de similaridade do artista, o valor do cálculo de similaridade do nome do artista.

Falta calcular a segunda parte da equação que está relacionada com os álbuns do artista. Nesta parte do algoritmo, é tido em consideração a informação disponível de cada álbum. Neste caso em particular, com exceção do álbum que se pretende inserir, os únicos detalhes presentes sobre os álbuns são os seus nomes, pelo que estas serão as únicas caraterísticas utilizadas para efetuar a segunda parte do cálculo do artista.

Um exemplo prático do porquê de ter de ser feito esta consideração é quando um mesmo álbum está associado a dois artistas. Ora, se estamos a tentar encontrar o artista, então não nos podemos basear apenas no álbum que está a ser inserido, mas também nos outros álbuns que lhe pertencem, para ajudar a perceber qual é o artista no sistema mais similar.

À semelhança do artista, um álbum também é caraterizado pelos seus nomes, porém também é pelas músicas que lhe estão associadas. Para o cálculo de similaridade do álbum é efetuada a seguinte fórmula:

$$\text{similaridade\_album} = 0.5 * (\text{similaridade\_do\_nome} + \text{similaridade\_caract\_musicas}) \quad (3)$$

Como foi mencionado anteriormente, para cada artista armazenado no sistema é extraído o conjunto de nomes que lhe está associado bem como os álbuns. Para efetuar o cálculo do álbum também vai ser necessário informação semelhante. Neste caso, o conjunto dos nomes associados para cada álbum, bem como o conjunto de músicas associadas a cada.

Para cada álbum extraído é verificada a similaridade dos seus nomes, comparando-o com o nome do álbum a inserir (fornecido pelo Local Agent). No caso das músicas é efetuada uma iteração para cada elemento do conjunto de músicas extraídas de cada álbum do Music Service e cada elemento do conjunto de músicas fornecidas pelo Local Agent. Esta comparação é baseada nas caraterísticas que lhes estão diretamente associadas, ou seja, o seu nome e o seu tempo de duração. O cálculo de similaridade de uma música é representado pela seguinte fórmula:

$$\text{similaridade\_musicas} = 0.8 * \text{similaridade\_do\_nome} + 0.2 * \text{similaridade\_da\_duracao} \quad (4)$$

No final da iteração das músicas associadas a um álbum, obtêm-se para cada música do Local Agent a música mais semelhante. De seguida, todos os valores máximos de similaridade encontrados para cada música são somados, obtendo o resultado referente à segunda parcela do cálculo de similaridade para o álbum, fórmula 3, que é normalizado entre o valor 0 e 1.

De igual modo, no final da iteração dos álbuns dos artistas obtém-se o álbum do Music Service que mais se assemelha ao álbum do Local Agent. Como foi feita uma iteração sobre todos os álbuns do artista foi ainda possível obter a segunda parcela do cálculo do artista 2 referente à similaridade dos álbuns.

Assim, obtêm-se todos os valores necessários para calcular a similaridade do álbum (fórmula 3) e, por conseguinte, a similaridade do artista (fórmula 2), que permitem calcular o valor de similaridade final, ou seja, do par artista + álbum (fórmula 1.

No final da iteração sobre os artistas, álbuns e músicas presentes no sistema do Music service, e na comparação com os dados a inserir no sistema fornecidos pelo Local Agent obtém-se a seguinte informação:

- valor máximo de similaridade para os elementos, neste caso o par artista e álbum. Sendo que para além do valor é guardado o identificador artista e do álbum, bem como as suas caraterísticas;
- valor máximo de similaridade para o artista. Onde também é guardado o seu identificador e as suas caraterísticas. De notar que este artista pode ser diferente do artista do ponto anterior;
- valor máximo de similaridade para o álbum. Onde também é guardado o seu identificador e as suas caraterísticas, e mais uma vez, o álbum com o valor máximo de similaridade pode ser diferente ao álbum encontrado no primeiro ponto;
- O valor máximo de similaridade para cada gênero do Local Agent, para além da identificação do gênero do Music Service correspondente.

Recorde-se o segundo exemplo apresentado na subsecção 4.8 na 3ª Fase, onde se vai fazer a comparação com os elementos B2, A2 (com as músicas M1 e M2). O objetivo é encontrar a similaridade dos elementos, ou seja, se o par artista e álbum já existem, que corresponde à fórmula 1.

Para tal, é necessário, primeiro calcular tanto o valor de similaridade do artista bem como o do álbum. Começando pelo cálculo do artista, inicialmente é preciso calcular a similaridade do nome, ou seja, a similaridade dos pares *pink floyd* mais *pink floyd* e *pink floyd* mais *pink floid*. Aplicando o algoritmo de similaridade de nomes, que é baseado na distância de Levenshtein, obtém-se o valor para o primeiro par de 1 e para o segundo 0,9. Ficando então com o valor máximo de 1 associado ao nome com o identificador N1.

Falta assim apenas uma parte, uma que é referente à similaridade dos álbuns. No entanto, este valor só é conseguido após se efetuar a iteração sobre os álbuns do artista que se está a analisar, no exemplo demonstrado, só após serem analisados os álbuns com os identificadores A1, A2 e A3.

Para calcular a similaridade de um álbum é necessário recorrer à fórmula 3. Nesta é feito um cálculo de similaridade do nome, semelhante ao que foi feito no artista, entre os pares *animals* mais *dark side of the moon* e *animals* mais *the dark side of the moon*, obtendo-se os valores de 0,14 e 0,12.

Olhando para a fórmula 3 resta calcular a similaridade das músicas a partir da fórmula 4. Nesta é comparada a similaridade das suas caraterísticas com uma maior ênfase no nome. Por exemplo para a música com o identificador M1 o valor é de 0,2 ao passo que para a música com o identificador M2 é de 0,18, quando comparadas com as caraterísticas *pigs on the wing (part I)* e 1'26".

Após extrair os valores de similaridade máximo para cada uma das músicas a inserir, 0,2 (*pigs on the wing part I*) e 0 (*dogs*), é efetuada a soma dos mesmos, obtendo o valor de 0,2. Como esta soma poderá ser maior do que 1, é feita uma normalização de maneira a colocar o valor entre 0 e 1. Neste caso em concreto após a normalização o valor é de 0,1, correspondente à similaridade das músicas do álbum A2.

Desta forma o valor de similaridade com o álbum A2 será de 0,12 (0,14 * 0.5 + 0,1 * 0.5). Assumindo que o valor de similaridade com o álbum A1 e o álbum A3 serão menores que o valor de similaridade alcançado com A2, encontramos o valor de similaridade máximo do álbum, que é de 0,12.

Para acabar o cálculo de similaridade do artista falta calcular a similaridade dos álbuns, que foi sendo calculada durante a iteração. Ou seja, a similaridade do álbum A1, A2 e A3 com a informação dos restantes álbuns que é fornecida: *The dark side of the moon*, *The wall* e *Wish you were here*. Nesta fase, vão ser só utilizados a similaridade dos nomes entre entre todos os álbuns.

Após aplicar o algoritmo de distância de Levenshtein obtém-se os seguintes valores:

- 1 para o álbum com o nome *The dark side of the moon* relacionado com o álbum com o identificador A2;
- 1 para o álbum com o nome *The wall* relacionado com o álbum com o identificador A1;
- 1 para o álbum com o nome *Wish you were here* relacionado com o álbum com o identificador A3;

Assim, no final da normalização, que é aplicada com a mesma estratégia que foi utilizada para o cálculo das músicas, o valor obtido para a similaridade de todos os álbuns do artista é de 0,79.

Desta forma, temos também concluído o cálculo do artista com o valor de 0,9 (1 * 0.5 + 0,79 * 0.5). Com os valores calculados para a similaridade do artista e a similaridade de cada álbum, obtém-se que o valor de similaridade do conjunto, artista mais o álbum com o maior valor de similaridade, é de 0,41 (0,9 * 0,35 + 0,12 * 0,65). Em suma, obtém-se os seguintes resultados:

- valor máximo de similaridade para os elementos: 0,41;
- valor máximo de similaridade para o artista: 0,9;
- valor máximo de similaridade para o álbum: 0,12.

Estes serão os valores alvos de análise para a segunda parte da terceira fase do processo de inserção de informação.

Cálculo do Gênero

Devido a razões anteriormente referidas, o gênero não entra na equação do cálculo da similaridade dos nomes. No entanto, é necessário efetuar uma procura no Music Service para saber se o gênero já existe ou não. Neste caso em concreto será único e exclusivamente utilizado o nome para diferenciar entre os gêneros.

Para o gênero é utilizada uma estratégia simples onde apenas se efetua um cálculo de similaridade sobre os nomes. Para tal é extraído os identificadores dos gêneros dos Music Service, bem como os nomes associados que lhes estão associados. Em seguida, para cada elemento dos gêneros a introduzir é corrido o algoritmo de similaridade dos textos e obtêm-se o valor máximo de similaridade. Para além deste valor é ainda retornado o gênero do Music Service correspondente.

Por exemplo, pegando na estrutura da figura 13, se num conjunto de dados a inserir está presente a seguinte informação sobre um gênero: *roock*. O que será executado pelo sistema é uma extração dos gêneros bem como as suas caraterísticas, e é efetuado o cálculo de similaridade para cada uma, obtendo-se os seguintes valores:

- 0 quando comparado com o gênero cujo identificador é G1 e G3;
- 0,8 quando comparado com o gênero cujo identificador é G2, com o nome *rock*.

Analisando os valores obtidos, facilmente se identifica que o valor máximo alcançado foi de 0,8 para o gênero com o identificador G2, sendo este o valor utilizado para a análise da existência do gênero na segunda parte da terceira fase do processo de inserção de informação.

3ª Fase, 2ª Parte - Inserção de Informação

A inserção de novos elementos vai se basear no valor de similaridade obtido para cada elemento, sendo que:

- Se o valor for igual ou superior a 0.63[1] significa que o artista e o álbum já se encontram no Music Service;
- Caso contrário ainda não existe o par na base de dados.

Quando o valor de similaridade é igual ou superior a 0.63, ou seja 63% de similaridade, não serão criados nem um novo artista nem um novo álbum porque já foi provado que os mesmos já estão representados na base de dados. Porém, podem sempre ter novas caraterísticas que devem de ser associadas aos elementos. No caso do nome:

- Se corresponder exatamente a um dos nomes associados ao artista ou álbum, então só é aumentado o número de ocorrências para o respetivo identificador do nome associado ao respetivo elemento;
- Se for diferente de todos os nomes associados ao elemento, então é criada uma nova ligação para associar esta nova designação.

Porém, quando o valor de similaridade é inferior a 0.63 significa que não existe o par artista e álbum. No entanto, não se pode descartar a ideia de que ambos não existem no Music Service.

Voltando a recordar o segundo exemplo apresentado na subsecção 4.8 na 3ª Fase, cuja foi feita uma simulação dos valores alcançados, sendo que o par artista + álbum (B2 + A2) do sistema mais similar teve o valor de 0,41. Ora, 0,41 < 0,63 pelo que se prova que o par artista + álbum a inserir (*pink floyd* e *animals*) ainda não existe.

Porém, o nível de similaridade encontrado para o artista foi de 0,9 ao passo que para o álbum foi de 0,12. Estes valores serão alvos de análise para verificar se já existe um artista ou álbum no sistema similar ao artista ou álbum que se pretende inserir.

Desta forma, serão considerados outros dois dados que foram obtidos durante a procura: o valor máximo de similaridade do artista, bem como o artista correspondente, e o valor máximo de similaridade do álbum, bem como o álbum correspondente. Em ambos os casos, o valor é analisado da mesma forma e tomadas as seguintes medidas:

- Se o valor for igual ou superior a 0.63, quer dizer que o elemento já existe no Music Service e que se deve de acrescentar ou atualizar as suas caraterísticas;
- Se o valor for inferior a 0.63, então o elemento não existe e deve de ser adicionado um novo ao Music Service com as caraterísticas providenciadas pelo Local Agent.

No exemplo que se está a seguir conclui-se que o artista já existe, já que 0,9 > 0,63, e o artista correspondente possui o identificador B2. Ao passo que o álbum não existe uma vez que o maior valor de similaridade para o álbum foi de 0,14 que é menor que 0,63.

1 A obtenção deste valor, bem como os outros que serão mencionados, teve por base testes que serão mostrados na sub-secção 4.8.4

No caso das músicas a sua adição é influenciada pela inserção de um novo álbum ou não. Como uma música está associada a um e apenas um álbum então se for adicionado um novo álbum, todas as músicas do Local Agent são adicionadas ao Music Service (o que se procederia com o exemplo apresentado anteriormente). Caso seja detetada a existência do álbum, é tido em conta o valor de similaridade para cada uma das suas músicas, e a inserção de uma nova música é determinada tendo em conta este valor, sendo que se:

- O valor de similaridade é 1, significa que a música já existe com exatamente as mesmas caraterísticas, pelo que só deve de ser aumentada o número de ocorrências para as mesmas (neste caso constituídas pelo par nome e duração);
- Se o valor for igual ou superior a 0.7 e inferior a 1, significa que a música já existe, no entanto são lhes associadas umas caraterísticas novas;
- Se o valor for menor que 0.7 significa que não há uma música do Music Service pertencente aquele álbum que seja similar, pelo que deve de ser acrescentada uma música nova.

Se a música possuir uma partição de álbum, então esta deve de ser criada para aquele álbum e associada à música, caso não exista no Music Service. Se existir, é apenas associada à música em questão.

No caso do gênero, como mencionado outrora, é tido em conta o seu nome e executado um cálculo para obter o valor de similaridade. Sendo que pode ocorrer um dos três cenários:

- Se o valor for 1, significa que é o mesmo gênero inequivocamente, e que já possui o nome associado pelo que vai ser aumentado o número de ocorrências;
- Se o valor for maior que 0.6 e menor que 1, os nomes dos gêneros são suficientemente semelhantes para serem associados ao mesmo gênero, no entanto é associada uma nova designação para o gênero do Music Service;
- Se o valor for menor que 0.6 então os nomes são distintos o que faz com que se chegue à conclusão de que o gênero ainda não existe no Music Service e então é criado um novo.

No caso de estudo que tem vindo a ser explorado, o valor máximo de similaridade alcançado foi de 0,8 correspondendo ao gênero com o identificador G2. Como 0,6<0,8<1, então seria adicionado uma nova designação ao gênero G2.

No final da inserção de todos os elementos musicais, quer seja pela criação de um novo ou pela atualização de existentes, são efetuadas as associações necessárias. A ligação entre o gênero - artista e artista - álbum. A ligação entre álbum - música é logo efetuada aquando da criação da música (porque uma música só pertence a um álbum).

Como são realizadas várias inserções em momentos dispares, pode sempre acontecer um problema e a execução da inserção não ser completa.

Nestes casos, a base de dados ficaria incoerente, uma vez que teria informação que não estaria completa. Para prevenir esta situação e como está a ser utilizada uma base de dados SQL são utilizadas transações para inserir a informação.

Desta forma, se um erro for detetado é efetuado um *rollback* e a base de dados fica com um estado anterior à inserção da informação.

Quando isto acontece o cliente é avisado do sucedido, neste caso o Local Agent, e reenvia a informação, sendo que esta terá de passar novamente por todo o processo de procura e inserção. No entanto, esta situação raramente acontecerá pelo que não se deve de manifestar nem afetar a experiência de utilização dos diferentes utilizadores.

Identificação do Proprietário da Informação

Para terminar a inserção da informação, falta só estabelecer a ligação para o Local Server correspondente (representação do Local Agent no Music Service) de forma a indicar a quem é que o conteúdo pertence. Isto pressupõe que seja criada uma ligação entre o identificador do Local Server e o identificador do elemento musical correspondente, para cada elemento que o Local Agent possua. Esta estratégia aumenta a utilização do espaço de armazenamento, no entanto facilita o acesso do conteúdo e validação de que se um Local Agent o contêm ou não, quando é para efetuar a apresentação para o utilizador final.

Caso não se optasse por esta estratégia, então quando fosse para ser apresentada a informação ao utilizador final teria de se interrogar os respetivos Local Agents para saber se continham o conteúdo, o que aumentaria o tempo de resposta e que tornaria a utilização do sistema inviável.

Apesar de ser utilizado mais espaço de armazenamento, com o par de identificador do Local Agent e do identificador do elemento, é uma estratégia viável pela rapidez na resposta ao utilizador final.

4.8.4 *Determinação do Valor Mínimo Ótimo de Similaridade*

Nas secções anteriores foi explicado o algoritmo de procura que avalia para uma nova informação se a mesma já existe no sistema. Como o algoritmo de procura é baseado num cálculo de similaridade entre dois elementos, então foi necessário estabelecer um valor mínimo que indique que o conteúdo já existe, mas que potencialmente está representado de uma maneira distinta. Existem quatro valores principais para definir a similaridade entre conteúdo, sendo eles:

SIMILARIDADE DO CONJUNTO: neste caso é considerada a informação como um todo, cálculo do artista mais o cálculo do álbum;

SIMILARIDADE DO ARTISTA: apenas considerado o cálculo do artista e serve para comparar o artista a inserir com os artistas já existentes;

SIMILARIDADE DO ÁLBUM: apenas considerado o cálculo do álbum e é utilizado para comparar a existência individual do álbum;

SIMILARIDADE DAS CARATERÍSTICAS DA MÚSICA: apenas considerado o cálculo da música, e à semelhança dos anteriores, serve para comparar outras caraterísticas das músicas.

Para determinar qual o valor ótimo de similaridade foi realizado dois conjuntos de testes, um primeiro para determinar os três primeiros valores, uma vez que estão todos relacionados, e um segundo apenas para determinar qual o valor mínimo para que duas músicas sejam consideradas semelhantes.

Testes para os valores de similaridade do conjunto, artista e álbum

O sistema foi carregado com meta-informação existente no Spotify e adquirida através da API pública disponibilizada pela entidade, Spotify Developers [65]. Foi extraída informação referente a 100 músicas, 79 álbuns e 55 artistas.

Foram definidos testes unitários, que podem ser consultados no anexo B, com a simulação de inserção de informação no sistema. Em cada teste é avaliado a tomada de decisão do algoritmo de procura do conteúdo para inserção sobre o artista e o álbum, ou seja, se o elemento já existe, se o elemento existe, mas com uma informação diferente ou se o elemento é novo.

Na figura 14 está representado um gráfico com a percentagem de acerto para diferentes níveis de similaridade. Um valor mais próximo de 0 indica que os elementos são dispares ao passo que um valor mais próximo de 1 indica uma maior semelhança entre o conteúdo. A partir do gráfico, o valor ótimo para a similaridade do conjunto, artista e álbum é 0.63, ou seja, quando a similaridade é igual ou superior a 63%.

Não são considerados todos os valores possíveis, uma vez que íamos ter algumas falácias com os testes realizados. Os testes são uma pequena proporção de um grande universo que permite avaliar vários acontecimentos possíveis durante a inserção. Valores abaixo de 50% ou muito próximo dele, faz com que todas as caraterísticas consideradas no cálculo do artista e do álbum deixem de ter em conta o seu peso no cálculo, uma vez que estes valores são muito pouco indicativos da similaridade de conteúdo. Desta forma, os testes começaram com o valor de 55%.

Todos os valores falham num teste, que é quando se tenta inserir uma informação sobre um novo artista que interpreta uma música do álbum de um outro artista. O sistema não está

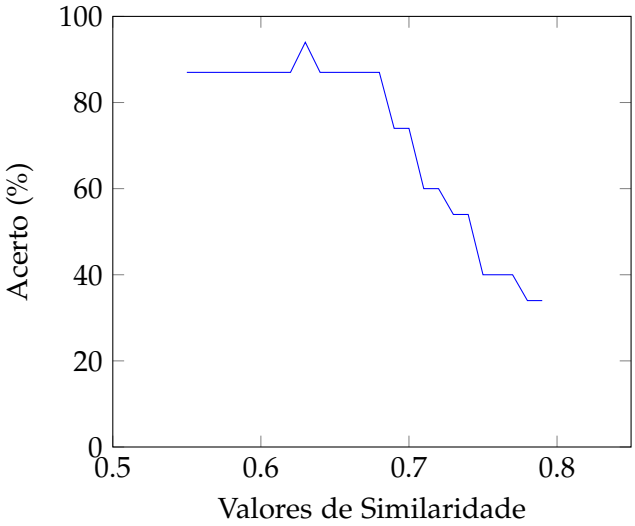


Figura 14: Resultado dos testes para os valores de similaridade do conjunto, álbum e artista

desenhado para representar este tipo de informação, é uma limitação que será referenciada adiante.

Testes para o valor de similaridade da música

No caso da música, como as caraterísticas avaliadas são o nome a duração, foi criado um conjunto de testes com dados não reais (que podem ser consultados em B), que permitem avaliar a diferença entre os nomes e a duração, no sentido de encontrar um valor que melhor avalie a similaridade entre as caraterísticas.

Na figura 15 está representado um gráfico com a percentagem de sucesso para cada valor diferente, sendo que o que apresentou melhores resultados foram o 0.70 e 71, ou seja, quando as caraterísticas das músicas são semelhantes entre 70% e 71%, sendo que o valor escolhido foi 70%.

De salientar, que as falhas apresentadas advêm de diferenças significativas do nome das músicas, quando por exemplo é trocada a ordem de algumas letras, como por exemplo, *love* e *olev*, mas neste caso mesmo para um ser humano fica difícil de entender que as duas palavras têm o mesmo significado. No entanto, quando é trocada uma letra ou aparece uma letra a menos ou várias a mais ou há a omissão de uma conjunção, por exemplo "de", os valores ótimos acertam nestes casos.

Nos casos em que uma música é muito similar tanto no nome como na duração com uma outra o algoritmo pode determinar uma falsa existência da música no sistema. No entanto, estas situações serão esporádicas, uma vez que se vai limitar a comparação de músicas dentro de um mesmo álbum (e não é esperado que existem muitas músicas com nome e duração semelhante).

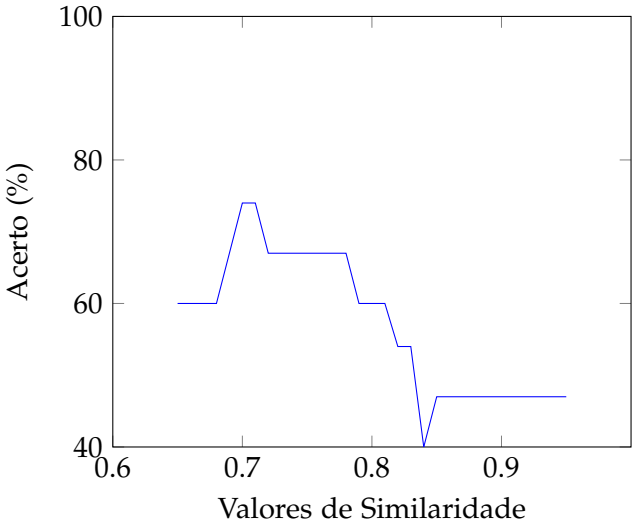


Figura 15: Resultado dos testes para os valores de similaridade das caraterísticas da música

4.8.5 *Eficiência em Memória*

O algoritmo que analisa a informação presente no sistema tem de percorrer todo o seu conteúdo. Ora carregar toda a informação para memória seria ineficiente e à medida que os dados fossem crescendo seria impensável fazê-lo, uma vez que não existiria memória suficiente para albergar toda a informação.

Desta forma, é utilizada uma técnica de *streaming* para extrair os dados da base de dados. São utilizadas ferramentas, tanto para Node como para Java, que permitem a obtenção de um grande número de informação, mas sem ser carregada toda de uma vez. Sendo apenas guardado em memória uma parte de maneira a não ultrapassar a capacidade da mesma.

4.8.6 *Limitações*

Este algoritmo de procura pressupõe que nenhuma informação é acrescentada ao sistema por parte de mais nenhum servidor privado. Este facto limita a que apenas possa ser atendido um pedido de inserção em cada momento o que diminui o tempo de resposta para os diferentes Local Agent e pode levar a que estejam vários pedidos em espera para inserção de conteúdo.

No entanto, esta limitação não afeta a experiência de utilização do cliente da aplicação móvel. Este apenas não vai poder aceder ao conteúdo logo de imediato que o Local Agent é registado, uma vez que o processo de inserção pode ser demorado.

Como o processo de *upload* de informação de um Local Agent será, em princípio, demorado pode ocorrer erros e nem todo o conteúdo fique disponibilizado. Neste caso, existem

mecanismos para detetar esta situação o que permite que numa reinício do programa a restante informação seja carregada.

4.9 PESQUISA DE CONTEÚDO

Um utilizador, da aplicação móvel, tem a possibilidade de reproduzir conteúdo disponibilizado pelos Local Agents desde que tenha permissões de acesso aos mesmos. Para realizar esta ação tem de conseguir aceder ao mesmo remotamente. Existem duas maneiras distintas de o fazer:

- Navegação do conteúdo, através dos diferentes níveis, de um Local Agent;
- Pesquisa por termos, através de texto.

A primeira ação é mais utilizada para exploração de conteúdo, uma vez que o utilizador tem a possibilidade de descobrir que tipo de conteúdo o Local Agent está a disponibilizar, navegando entre os diferentes níveis. No entanto, só pode realizar esta ação para um Local Agent de cada vez.

Apesar de esta ação ser a ideal para quando o utilizador não conhece o conteúdo do Local Agent e pretende explorá-lo de modo a ficar com uma ideia do que pode encontrar, na maioria das situações, o utilizador pode já ter uma ideia do que pretende reproduzir e só a disponibilização deste método para encontrar conteúdo tornaria a experiência de utilização negativa.

Quando o utilizador já tem a ideia do conteúdo que pretende reproduzir, com esta estratégia, ele tem de saber uma série de informações sobre a música, como gênero, artista, álbum, partição do álbum (no caso de existir) e ainda o nome da mesma. Em muitos dos casos, o utilizador apenas sabe o nome, ou uma parte do nome, da música ou que a mesma pertence a um certo artista. Nestes casos, seria penoso para o utilizador ter de andar a procurar de entre todo o conteúdo do Local Agent. Pior seria se aquele Local Agent não disponibilizasse o conteúdo necessário. A navegação do conteúdo é boa para determinados casos, porém não é possível obter diretamente o conteúdo.

Devido a este problema, o utilizador tem a possibilidade de efetuar uma pesquisa através de texto, onde pode efetuar uma pesquisa geral, apenas por música, álbum ou artista e ainda a possibilidade de procurar em todos os Local Agent que tem acesso ou algum em específico.

Uma vez que está a ser utilizado texto para encontrar o conteúdo a estratégia adotada baseia-se num algoritmo de similaridade entre textos, à semelhança do que é feito na procura de conteúdo no processo de inserção de nova informação. No entanto, esta pesquisa de conteúdo é mais simples, quando comparada com a estratégia de procura vista anteriormente, uma vez que a única caraterística a ser tida em conta vai ser o nome.

O algoritmo utilizado é o de Levenshtein + inclusão, utilizando implementação da biblioteca *fuzzball* [66] em Node, apoiando esta decisão nos resultados apresentados na pesquisa Development of a Client Application for Music Server Access through the Simple Network Management Protocol [59]. O utilizador tem a possibilidade de realizar as seguintes ações para a pesquisa:

- geral onde não especifica a que elemento as caraterísticas introduzidas pertencem;
- restringir a pesquisa apenas a artistas, álbuns ou músicas. Sendo que pode utilizar as permutações que bem entender;
- restringir em quais Local Agent quer efetuar a pesquisa de entre os quais tem acesso. No caso de ser omitido o Local Agent, então procurará no conteúdo de todos.

Para realizar a pesquisa nos conteúdos é necessário ter em conta dois cenários:

1. o utilizador restringiu a procura para apenas um dos elementos;
2. a pesquisa vai ser efetuada sobre mais do que um elemento.

O primeiro cenário é mais simples uma vez que só é necessário efetuar uma procura em apenas um elemento. É extraída uma lista dos nomes associados aos elementos que pertençam aos Local Agents que o utilizador tem acesso ou então ao Local Agent especificado. Para cada nome é aplicado o algoritmo de similaridade e obtido um valor que está compreendido entre 0 e 1, quanto mais perto de 1 maior é a similaridade entre os textos. São filtrados todos os resultados com um valor inferior a 0.6 e são todos ordenados por ordem decrescente de valor.

No final, são extraídos os elementos associados aos nomes e retornados ao utilizador por ordem decrescente de valor, onde o primeiro da lista tem uma maior probabilidade de ser o conteúdo que o utilizador pretende consultar ou reproduzir.

No segundo cenário é necessário efetuar uma junção dos resultados dos diferentes elementos. A extração dos nomes segue a mesma estratégia, no entanto são extraídas tantas listas quantos os elementos que tenham sido selecionados. No final para cada uma das listas são novamente excluídos os nomes cujo valor de similaridade seja inferior a 0.6 e ordenado por ordem decrescente de valor.

Após a conclusão desta etapa, é efetuada uma fusão do conteúdo filtrado e é novamente ordenado por ordem decrescente de valor. São extraídos os elementos associados aos nomes bem como as suas associações. Por exemplo, no caso do primeiro nome estar associado a um álbum, é extraído os artistas daquele álbum e ainda as músicas do mesmo.

Para cada conjunto de elementos é corrido o algoritmo de similaridade, nos casos em que ainda não há um valor para o texto em questão, e somados todos os valores. No final, é

efetuada uma ordenação por ordem decrescente e retornado o resultado ao utilizador. Sendo que na primeira opção aparecerá o conteúdo com maior probabilidade de ser o conteúdo que o utilizador quereria consultar ou reproduzir.

5

CONCLUSÕES

Os trabalhos descritos nesta dissertação foram desenvolvidos no sentido de definir e validar uma arquitetura duma plataforma de *streaming* de áudio a pedido que permitisse o acesso remoto através da Internet a coleções musicais privadas dos utilizadores e que são geridas e mantidas em sistemas computacionais nas suas redes locais. Os principais requisitos funcionais incluíam a utilização restrita de tecnologias, mecanismos e protocolos abertos de integração gratuita, a capacidade do sistema de proteger a privacidade das coleções e, ao mesmo tempo, garantir o respeito estrito de todos os direitos legais associados aos conteúdos musicais incluídos nessas coleções. Mais ainda, era desejável que a arquitetura garantisse as capacidades de reprodução remota por *streaming* áudio de coleções privadas já existentes, sem que houvesse necessidade da sua transferência para servidores remotos de sistemas *cloud* ou transferência e delegação da sua gestão a plataformas de *streaming* áudio atuais, em que, adicionalmente, o acesso está condicionado a um registo num determinado tipo de serviço.

O estudo das soluções atuais disponibilizadas pelas plataformas atuais de *streaming* existentes no mercado serviu para confirmar que estes sistemas não correspondem a todos os principais requisitos funcionais definidos nos objetivos já referidos e que era necessário definir uma arquitetura um pouco diferente para este novo tipo de plataforma.

O modelo administrativo e funcional desta nova arquitetura é a principal contribuição dos trabalhos desta dissertação. Para além dos requisitos funcionais mais óbvios para esta nova plataforma, pode salientar-se a integração dum mecanismo de *streaming* que possibilita dois modos de funcionamento, direto e indireto. O primeiro permitindo *streaming* com maior desempenho (menores atrasos) e completamente independente de servidores de *relay* áudio na Internet e o segundo permitindo, através desses servidores de *relay* de *streaming* áudio, um funcionamento mais seguro e com garantias maiores, não só de privacidade da rede local do utilizador bem como de interoperabilidade entre os dispositivos nas redes locais dos utilizadores (como, por exemplo, os Local Agents) e os dispositivos clientes de reprodução áudio remota na Internet usados por esses utilizadores. Mais ainda, o modelo automático, centralizado e contributivo de registo e gestão da meta-informação das coleções de todos os utilizadores da plataforma (implementado no Music Service) é uma mais valia

(a meta-informação de cada coleção individual deixa de ser precisa) pois acelera o processo de procura e escolha das músicas, facilitando também a implementação de sistemas de recomendações.

Para demonstrar a validade do modelo proposto foi implementado um protótipo de todo o sistema, incluindo o desenvolvimento de todos os seus componentes principais, tendo sido utilizada apenas tecnologia gratuita e aberta (*freeware & open-source*). Através do teste do protótipo foi possível concluir que a arquitetura permite uma gestão sustentável de todos os recursos associados aos componentes implementados para a rede local do utilizador bem como aos associados aos componentes implementados para funcionamento do serviço na Internet, tendo em conta um equilíbrio entre eficiência e replicação dos dados.

O modelo proposto segue um paradigma em que é fácil a comunidade poder desenvolver componentes em separado e acrescentar funcionalidades no futuro. Adicionalmente, o modelo teórico não tem limitações no suporte a quaisquer formatos para os ficheiros áudio das coleções dos utilizadores, nem limitações na qualidade do áudio suportada pelo *streaming* remoto. O próprio protótipo suporta as mais relevantes variantes dos formatos áudio mais importantes (mp3, FLAC, PCM, etc.) e a qualidade do *streaming* é apenas limitado pela qualidade dos formatos dos ficheiros originais em cada coleção.

Mais ainda, o sistema oferece compatibilidade com diversos formatos de áudio (exemplo: mp3) que abrange uma grande percentagem do mercado existente, permitindo que qualquer pessoa possa usufruir de toda a sua coleção musical, ou em grande parte da mesma.

O protótipo desenvolvido incluiu também uma aplicação cliente reprodutora (User Agent) para utilização em dispositivos móveis Android. Os testes desta aplicação demonstraram que é possível a construção deste tipo de aplicação com dispositivos móveis genéricos com acesso à Internet, sem quaisquer exigências relevantes de recursos.

As principais limitações do protótipo atual prendem-se com questões do processo de construção de *software* em ambiente académico, com recursos humanos limitados em quantidade e em tempo. A implementação otimizada dum sistema completo desta natureza não se coaduna com o desenvolvimento em equipas dum único elemento num período temporal relativamente limitado. Talvez a principal limitação do protótipo tenha a ver com as limitações da implementação do sistema de agregação e gestão da meta-informação das coleções de todos os utilizadores. Este sistema é relativamente complexo e no seu desenvolvimento, apesar de tupo e no contexto dos trabalhos de desenvolvimento do software dos componentes do protótipo, demorado, não foram usadas tecnologias avançadas para o tratamento da informação, como por exemplo, estratégias baseadas em metodologias de inteligência artificial.

Como conclusão final pode dizer-se que os objetivos mais relevantes dos trabalhos descritos e discutidos nesta dissertação foram atingidos plenamente. A principal limitação que se pode apontar é não especificação dum modelo de informação formal para a representação

e manipulação da meta-informação no Music Service e no Local Agent. Isto permite, atualmente, que possam ser implementados modelos de informação distintos para os dois componentes, o que dificulta a agregação da meta-informação das coleções privadas no Music Service. Aliás, este foi um dos problemas encontrados durante o desenvolvimento do software do Music Service e do Local Agent, uma vez que as suas implementações incluem modelos de representação e codificação da meta-informação ligeiramente diferentes. Da mesma forma teria sido importante especificar formal e detalhadamente as unidades de dados protocolares transferidas em todas as interações possíveis entre os componentes da arquitetura quando são utilizados protocolos aplicacionais próprios.

TRABALHO FUTURO

Para além das principais limitações já referidas neste capítulo, e que têm a ver com falta de completude na especificação formal de alguns aspetos da arquitetura proposta, sobretudo no que concerne às interações protocolares entre os seus componentes e que implementam processos de controlo (e não processos de *streaming* do áudio propriamente dito), podem encontrar-se mais facilmente limitações que se prendem com a não utilização de estratégias e mecanismos otimizados na implementação dos vários componentes do protótipo. Por exemplo, não são utilizados mecanismos de paralelização de processos nem qualquer metodologia formal de inteligência artificial.

Portanto, é natural que se sugiram como primeiras etapas de trabalho futuro:

- Completar uma especificação formal unificadora do modelo de informação para agregação da meta-informação em todos os componentes da plataforma que o necessitem;
- Completar uma especificação formal de todas as unidades de dados protocolares trocadas entre todos os componentes da arquitetura;
- Completar uma especificação formal dos protocolos aplicacionais utilizados e que não sejam utilizados nas tecnologias de *streaming* de áudio já existentes.

Por outro lado, em termos de melhorias do protótipo desenvolvido, podem recomendar-se as seguintes evoluções:

- Refazer o *software* do componente Local Agent e do componente Music Service por forma a que incluam uma implementação estrita do modelo de informação, da serialização das unidades protocolares de dados e das interações dos protocolos aplicacionais de acordo com as especificações acima referidas;
- Melhoria do sistema de cache utilizada no Music Service também por forma a permitir intercalar o *streaming* do conteúdo com meta-informação;

- Implementação do modo de servidor virtual no Local Agent;
- Desenvolver uma aplicação cliente para reprodução áudio remota para dispositivos com sistema operativo iOS e sistema operativo Windows/MacOS por forma a aumentar a compatibilidade do sistema protótipo num dos aspetos que mais facilmente tem impacto na utilização deste tipo de plataformas.

Por fim, resta realçar a vontade de tentar escrever e publicar um artigo científico numa revista internacional e que apresente e discuta as contribuições mais relevantes deste trabalho.

REFERÊNCIAS

[1] *Real-time Transport Protocols*. URL: https://en.wikipedia.org/wiki/Real-time_Transport_Protocol.

[2] *Real Time Streaming Protocol (RTSP)*. URL: https://searchvirtualdesktop.techtarget.com/definition/Real-Time-Streaming-Protocol-RTSP.

[3] *DLNA*. URL: https://www.dlna.org/.

[4] *Universal Plug and Play*. URL: https://en.wikipedia.org/wiki/Universal_Plug_and_Play.

[5] A. S. D. Corporation, C. Systems, I. Corporation, M. Corporation, N. Corporation, P. Electronics e S. Eletronics. *UPnP Device Architecture 1.0*. Rel. téc. UPnP Forum, abr. de 2008.

[6] A. Donoho, B. Roe, M. Bodlaender, J. Gildred, A. Messer, Y. Kim, B. Fairman e J. Tourzan. *UPnP Device Architecture 2.0*. Rel. téc. UPnP Forum, fev. de 2015.

[7] J. Ritchie, T. Kuehnel, W. van der Beek e J. Kang. *UPnP AV Architecture:2*. Rel. téc. UPnP Forum, dez. de 2010.

[8] *Difference between UPnP v1 / UPnP v2 / MiniDLNA*. Abr. de 2014. URL: https://support.asustor.com/index.php?/Knowledgebase/Article/View/138/0/difference-between-upnp-v1--upnp-v2--minidlna.

[9] J. Ritchie e T. Kuehnel. *UPnP AV Architecture:1*. Rel. téc. UPnP Forum, jun. de 2002.

[10] *UPnP vs UPnP2: What's the difference between upnp and upnp2?* URL: https://community.ui.com/questions/UPnP-vs-UPnP2-Whats-the-difference-between-upnp-andupnp2/1d11eb63-3a65-4bd0-be89-30b399a0fc22.

[11] *Protocolo de mapeamento de portas NAT*. URL: https://pt.wikipedia.org/wiki/Protocolo_de_mapeamento_de_portas_NAT.

[12] A. Green. *What is UPnP & Why is it Dangerous?* Mar. de 2018. URL: https://www.varonis.com/blog/what-is-upnp/.

[13] V. Lortz e M. Saaranen. *DeviceProtection:1 Service*. Rel. téc. UPnP Forum, fev. de 2011.

[14] C. Ellison. *DeviceSecurity:1 Service Template*. Rel. téc. UPnP Forum, nov. de 2003.

[15] J. List. *UPnP, vulnerability as a feature that just won't die*. Jan. de 2019. URL: https://hackaday.com/2019/01/14/upnp-vulnerability-as-a-feature-that-just-wont-die/.

[16] J. Tidy. *PewDiePie hackers take over Google smart TV systems*. Jan. de 2019. URL: https://www.bbc.com/news/technology-46746592.

[17] *DLNA Guidelines*. Jun. de 2016. URL: https://spirespark.com/dlna/guidelines/.

[18] M. Ansaldo. *Everything you need to know about DLNA: The de facto home-entertainment network standard*. Mar. de 2015. URL: https://www.techhive.com/article/2020825/how-to-get-started-with-dlna.html.

[19] *O que é a função DLNA e como funciona?* URL: https://www.sony.pt/electronics/support/articles/00106319.

[20] *Writing profiles for DLNA devices*. Jul. de 2013. URL: https://forums.plex.tv/t/writing-profiles-for-dlna-devices/38060.

[21] *DLNA Guidelines - Technical Overview*. Jun. de 2016. URL: http://www.dlna.org/dlna-for-industry/technical-overview.

[22] *DLNA Certified Device Classes*. URL: http://web.archive.org/web/20101222205822/http:/www.dlna.org/digital_living/devices/.

[23] *HTTP*. URL: https://developer.mozilla.org/pt-BR/docs/Web/HTTP.

[24] *Digital Audio Access Protocol (DAAP)*. URL: http://daap.sourceforge.net/docs/index.html.

[25] *DAAPDocumentation.wiki*. URL: 2.https://code.google.com/archive/p/ytrack/wikis/DAAPDocumentation.wiki.

[26] *Portas TCP e UDP utilizadas pelos produtos de software Apple*. Set. de 2019. URL: https://support.apple.com/pt-pt/HT202944.

[27] *Bonjour Concepts*. 2013. URL: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/NetServices/Articles/about.html#//apple_ref/doc/uid/TP40002458-TPXREF107.

[28] *iTunes 4.5 Authentication Cracked*. URL: https://apple.slashdot.org/story/04/04/29/1554231/itunes-45-authentication-cracked.

[29] *Rhythmbox & daap with itunes 7, 8, 9, 10 doesn't work correctly*. Set. de 2006. URL: https://bugs.launchpad.net/banshee/+bug/62842.

[30] *What is Spotify?* URL: https://support.spotify.com/us/using_spotify/getting_started/what-is-spotify/.

[31] *Spotify Connect*. URL: https://support.spotify.com/us/listen_everywhere/on_speaker/spotify-connect/.

[32] B. Stephenson. *What Is Spotify And How Does It Work?* Jun. de 2019. URL: https://www.lifewire.com/what-is-spotify-4685829.

[33] *Audio Settings*. URL: https://support.spotify.com/us/using_spotify/system_settings/high-quality-streaming/.

[34] G. Kreitz e F. Niemela. "Spotify–large scale, low latency, P2P music-on-demand streaming". Em: *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*. IEEE. 2010, pp. 1–10.

[35] Ernesto. *Spotify Starts Shutting Down Its Massive P2P Network*. Abr. de 2014. URL: https://torrentfreak.com/spotify-starts-shutting-down-its-massive-p2p-network-140416/.

[36] *Spotify Removes Peer-to-Peer Technology From its Desktop Client*. URL: https://techcrunch.com/2014/04/17/spotify-removes-peer-to-peer-technology-from-its-desktop-client/\%20e\%20.

[37] E. Samuels e A. Muppalla. *The Evolution of Spotify Home Architecture*. Jun. de 2019. URL: https://www.infoq.com/presentations/evolution-spotify-arch/.

[38] *Documentação do Cloud Pub/Sub*. URL: https://cloud.google.com/pubsub/docs/.

[39] *Cloud Big Table*. URL: https://cloud.google.com/bigtable/.

[40] *Apple Music*. URL: https://www.apple.com/pt/apple-music/.

[41] *Pandora*. URL: https://www.pandora.com/.

[42] K. Wouk. *SiriusXM's $3.5 billion purchase of Pandora just might be music to our ears*. Jun. de 2018. URL: https://www.digitaltrends.com/home-theater/siriusxm-pandora-acquistion/.

[43] *Youtube Music*. URL: https://www.youtube.com/musicpremium.

[44] K. Wouk. *YouTube Music or Play Music? Google's Music Services Explained*. Jun. de 2018. URL: https://www.pcmag.com/news/youtube-music-or-play-music-googles-music-services-explained.

[45] S. Kieldsen. *6 things you need to know about YouTube Music*. Jun. de 2018. URL: https://www.stuff.tv/features/6-things-you-need-know-about-youtube-music.

[46] *Deezer*. URL: https://www.deezer.com/pt/.

[47] *Tidal*. URL: https://tidal.com/.

[48] *Amazon Music*. URL: https://music.amazon.com/.

[49] *MixCloud*. URL: ttps://www.mixcloud.com/about/.

[50] Mixcloud. *Moving toward a fair and sustainable streaming model for artists*. Jul. de 2019. URL: https://medium.com/mixcloud/moving-toward-a-fair-and-sustainable-streaming-model-for-artists-99050561a069.

[51] *iTunes*. URL: https://www.apple.com/pt/itunes/.

[52] *VOX Cloud Music*. URL: https://vox.rocks/loop-music-cloud-storage.

[53] *MyMusicCloud*. URL: https://eu.usatoday.com/story/tech/columnist/saltzman/2014/01/13/cloud-based-streaming-app-music/4458887/.

[54] *iTunes Match*. URL: https://support.apple.com/pt-pt/HT204146.

[55] *Double Twist*. URL: https://www.doubletwist.com/.

[56] *Double Twist - Cloud Player*. URL: https://www.doubletwist.com/cloudplayer.

[57] *Double Twist - Player*. URL: https://www.doubletwist.com/player.

[58] *Double Twist - Desktop*. URL: https://www.doubletwist.com/desktop.

[59] V. Coelho. *Development of a Client Application for Music ServerAccess through the Simple Network Management Protocol*. 2017.

[60] D. Oros. *What is Simple Network Management Protocol (SNMP)?* Jul. de 2016. URL: https://www.auvik.com/franklyit/blog/network-basics-what-is-snmp/.

[61] P. Benjamin. *Demystifying STRIDE Threat Models*. Dez. de 2018. URL: https://dev.to/petermbenjamin/demystifying-stride-threat-models-230m.

[62] *AudioTrack*. URL: https://developer.android.com/reference/android/media/AudioTrack.

[63] *Javazoom - Java Layer*. URL: http://www.javazoom.net/javalayer/javalayer.html.

[64] *FLAC library (Java)*. URL: https://www.nayuki.io/page/flac-library-java.

[65] *Spotify Developer API*. URL: https://developer.spotify.com/documentation/web-api/.

[66] *Fuzzball*. URL: https://www.npmjs.com/package/fuzzball.

A

MANUAL DE UTILIZAÇÃO

O sistema possui duas componentes onde existe uma interação com o utilizador. No Local Agent o responsável pelo conteúdo efetua a configuração inicial e na aplicação *mobile*, Client App, um utilizador com permissões de acesso tem a capacidade de reproduzir o conteúdo desejado.

A.1 LOCAL AGENT

A componente Local Agent é um programa que deve de ser instalado na máquina onde o conteúdo, que será disponibilizado, está localizado. Isto porque o programa vai ler o conteúdo a partir da diretoria indicada pelo utilizador, o que obriga a que o Local Agent e o conteúdo estejam os dois na mesma máquina. Isto não invalida que o utilizador pretenda utilizar um disco externo para guardar o seu conteúdo, no entanto tem de ser possível aceder dentro da máquina onde se encontra instalado o programa do Local Agent.

O Local Agent é disponibilizado numa imagem Docker para garantir que independentemente do sistema operativo da máquina do utilizador, este último consiga instalar e correr o programa. Desta forma o utilizador só tem de garantir que na sua máquina tem instalado um serviço de Docker, que pode ser instalado através da sua página oficial `https://www.docker.com/`

Após a instalação na máquina desta nova tecnologia, o utilizador deve de criar um ficheiro específico, *docker-compose.yml*, numa pasta da sua preferência. Neste ficheiro é onde o utilizador vai poder configurar o seu programa. O conteúdo do mesmo deve de ser semelhante a:

```
version: '2.1'

services:
  local_agent:
    container_name: local_agent
    image: jorgeoliv/local-agent
    environment:
      - DATA_DIR=YOUR_COLLECTION
```

```
    - HOSTNAME=YOUR_IP
    - PRIVATE_SERVER_NAME=YOUR_SERVER_NAME
    - PASSWORD=YOUR_PASSWORD
    - OWNER_NAME=YOUR_NAME
    - CONTENT_PASSWORD=YOUR_CONTENT_PASSWORD
    - ADMIN_PASSWORD=YOUR_ADMIN_PASSWORD
    - DATABASE_URL=jdbc:postgresql://local_agent_postgres:5432/localagent # only change it, if
        you have another db in postgres
    - DATABASE_USERNAME=localagent # change it if you wanna
    - DATABASE_PASSWORD=123456 # change it if you wanna
  volumes:
    - '/home/jorge/Desktop/fakeDir:/YOUR_COLLECTION'
    - './.localAgent:/localAgent/.localAgent/' # don't change this one
  depends_on:
    - local_agent_postgres
  ports:
    - 61000:61000
    - 51000-51100:51000-51100/tcp
  expose:
    - 61000
    - 51000-51100

local_agent_postgres:
  image: postgres:13.2
  container_name: local_agent_postgres
  environment:
    - POSTGRES_USER=localagent
    - POSTGRES_PASSWORD=123456
    - POSTGRES_DB=localagent
  volumes:
    - './data/postgres:/var/lib/postgresql/data'
  ports:
    - 54322:5432
```

Listing A.1: *docker-compose.yml*

Sendo que o utilizador tem de obrigatoriamente definir os seguintes campos:

DATA_DIR: é o caminho da diretoria base onde se encontra o conteúdo musical;

HOSTNAME: é o IP da máquina;

PRIVATE_SERVER_NAME: é o nome associado ao repositório. O que será apresentado na aplicação *mobile*. Deve de ser único, sendo que o programa dá um erro ao iniciar se o mesmo já estiver escolhido;

PASSWORD: é a palava-chave definida para realizar autenticação por parte do servidor privado no sistema central;

OWNER_NAME: é o nome do dono da coleção;

CONTENT_PASSWORD: é a palavra-chave que permite a que um utilizador da aplicação móvel aceda ao conteúdo deste repositório;

ADMIN_PASSWORD: é a palavra-chave que permite a que um utilizador da aplicação móvel aceda e gerencie o conteúdo deste repositório, bem como as definições associadas ao mesmo.

Após a criação deste ficheiro o utilizador só tem de colocar o programa a correr tendo em conta a documentação do Docker. Num sistema baseado em Linux pode ser feito executando o seguinte comando:

```
docker-compose -f docker-compose.yml up -d
```

Listing A.2: Comando em Linux para correr o *LocalAgent*

Após o programa iniciar, está preparado para poder aceitar novos pedidos e iniciar o *streaming* de conteúdo para os diferentes utilizadores da aplicação móvel com acesso ao repositório.

A.2 CLIENT APP

A componente Client App é uma aplicação *mobile* para dispositivos com o sistema operativo Android. Através da mesma um utilizador pode navegar pelos diferentes Local Agents registados no sistema e escolher qual o conteúdo que pretendem reproduzir, desde que tenham permissões de acesso ao mesmo.

Quando o utilizador inicia a aplicação pela primeira vez é lhe apresentado um ecrã que permite realizar a autenticação no sistema, onde deve de inserir o seu email de registo e a palavra-chave definida.

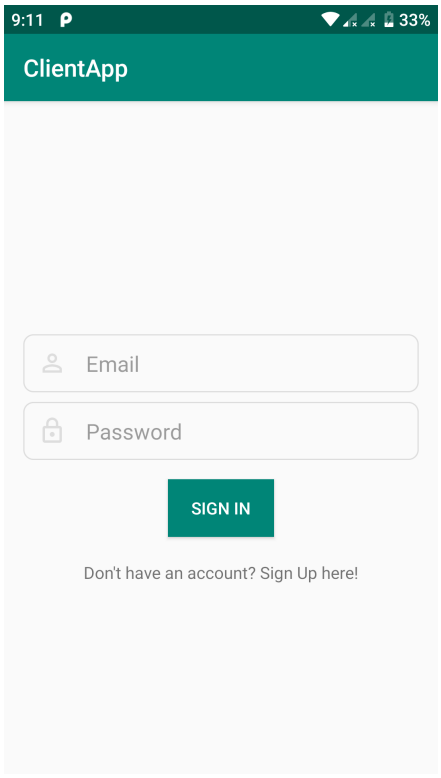


Figura 16: Ecrã de Autenticação

No caso de ainda não ter uma conta criada, tem sempre a possibilidade de se registar no sistema. Neste caso será redirecionado para outro ecrã onde deve de colocar o seu email e a palavra-chave.

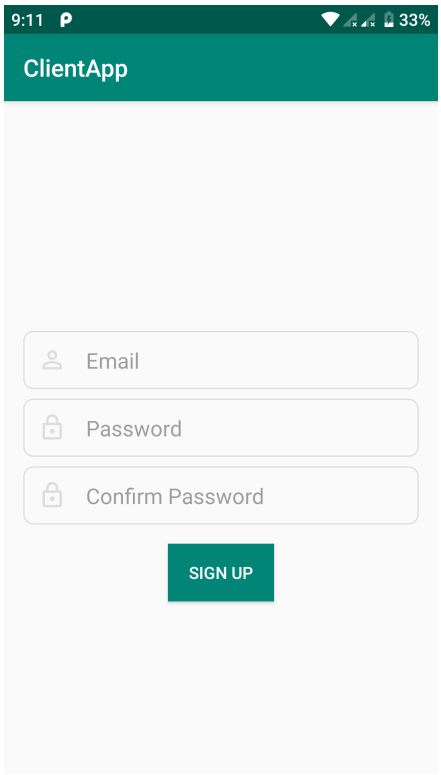


Figura 17: Ecrã de Registo

Em qualquer dos casos, se ocorrer um erro durante a autenticação ou registo, quer porque as credenciais não coincidam ou o email já se encontre utilizado, é sempre apresentado um erro explicito para que o utilizador consiga resolver a situação autonomamente.

A partir das próximas utilizações a autenticação estará gravada no dispositivo o que faz com que o utilizador não tenha de estar sempre a efetuar a autenticação quando abre a aplicação. Após, a efetuação da autenticação ou do registo com sucesso é redirecionado para o ecrã principal.

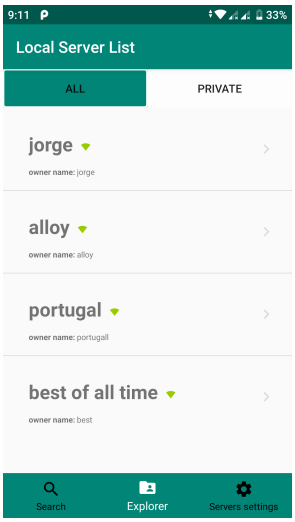


Figura 18: Ecrã Principal - Lista de Todos os Servidores Privados

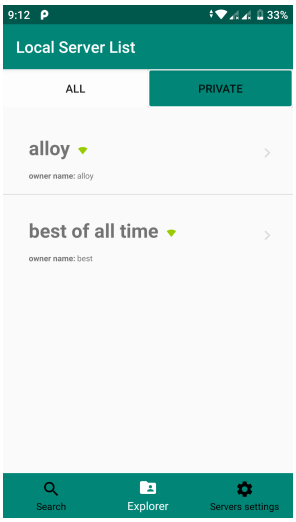


Figura 19: Ecrã Principal - Lista dos Servidores Autenticados

Neste são apresentados todos os local agents registados no sistema. São lhe apresentadas duas abas, uma com todos os servidores privados registados no sistema (local agents) e a outra com todos os servidores privados que o utilizador efetuou autenticação com sucesso.

Quando um é selecionado é lhe apresentado uma caixa de texto onde deve colocar a palavra-chave de acesso ao conteúdo. No caso da seleção de um local agent que já tenha realizado a autenticação no mesmo é logo redirecionado para o seu conteúdo. Assim, permite que o utilizador não tenha de estar constantemente a realizar a autenticação para um mesmo local agent.

Na Bottom Bar está presente um menu que permite navegar facilmente entre ecrãs na aplicação. A opção por defeito, *explorer*, já foi apresentada em cima. No entanto, existem mais dois menus *search* e *server settings*.

Ao selecionar um local agent permite que o utilizador navegue sobre o conteúdo do mesmo, como mostra na imagem seguinte.

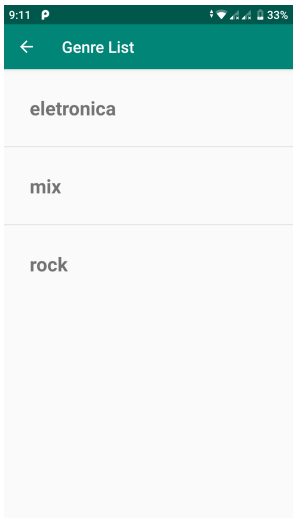


Figura 20: Ecrã Lista de Gêneros

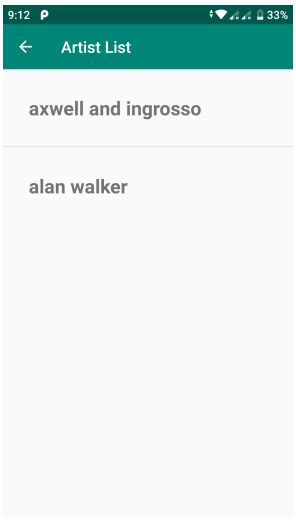


Figura 21: Ecrã Lista de Artistas

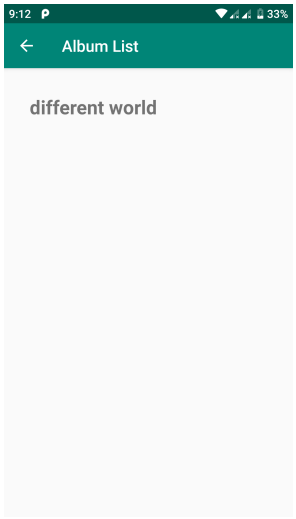


Figura 22: Ecrã Lista de Albuns

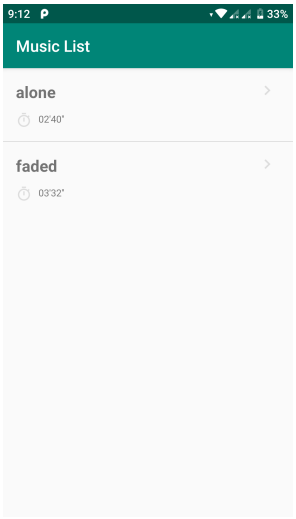


Figura 23: Ecrã Lista de Músicas

Num primeiro nível são apresentados os gêneros, em seguida os artistas, depois os álbuns, partições de álbuns (se existirem) e por fim a lista das músicas. Todos os níveis são devidamente identificados e têm uma apresentação similar, como é possível observar nas figuras 20 21 22 23

Quando o utilizador se encontra no último nível, a apresentação das músicas, ao selecionar uma dá-lhe acesso ao reprodutor de multimédia da aplicação.

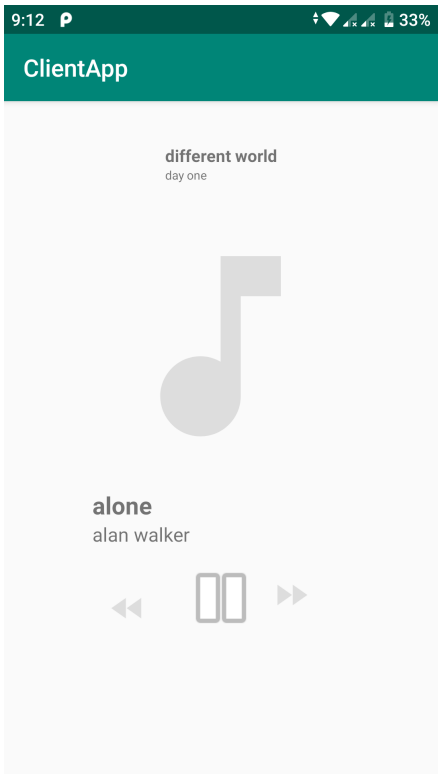


Figura 24: Ecrã de Reprodução Musical

Neste ecrã o utilizador tem a informação sobre todos os elementos associados à musica e ainda possui a capacidade de controlar toda a reprodução do conteúdo:

- Colocar em pausa;
- Retomar a reprodução;
- Avançar um intervalo de 30 segundos na música;
- Recuar um intervalo de 10 segundos na música;
- Parar a reprodução.

Sempre que o utilizador sai do ecrã a música é imediatamente parada, e no caso de ser selecionada a mesma música, esta vai ser reproduzida desde o início.

No caso de ser selecionado o menu de *search* é apresentado um ecrã que permite ao utilizador efetuar uma pesquisa por um conteúdo especifico através do nome.

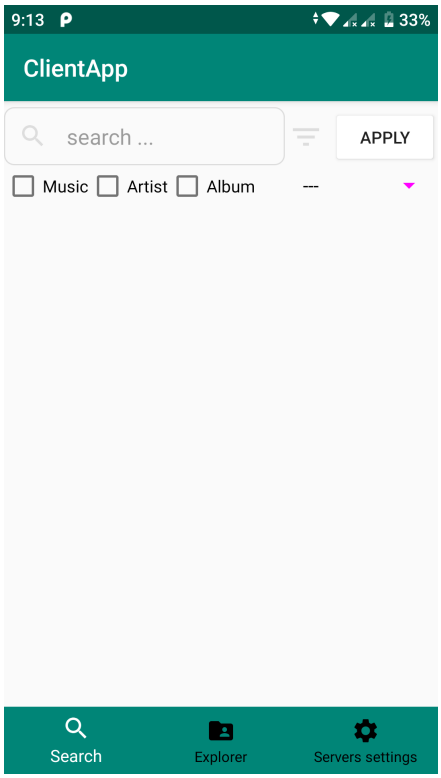


Figura 25: Ecrã de Procura de Conteúdo

O utilizador tem a capacidade de efetuar uma pesquisa genérica ou então apenas por músicas ou artistas ou álbuns e efetuar as permutações que pretender com estas opções. De salientar que, se selecionar as três caraterísticas em simultâneo o servidor vai atuar da mesma forma que se nenhuma caraterísticas não fosse selecionada, ou seja, efetua uma pesquisa genérica.

O utilizador tem ainda a possibilidade de procurar em todos os servidores privados que tem acesso ou selecionar apenas um deles, e pesquisa de conteúdo vai ser referente apenas ao selecionado.

Os resultados são apresentados numa lista semelhante à lista que aparece no ecrã da listagem das músicas referido anteriormente, e após a seleção de uma será redirecionado para o ecrã de reprodução musical, representado pela figura 23.

A opção "server settings", que se encontra na *bottom bar*, permite ao utilizador efetuar uma gestão dos seus servidores privados remotamente. Para tal, no momento da autenticação do servidor privado tem de colocar a *password* com permissões de acesso de administração. Nestes casos, todos os servidores com este tipo de acesso aparecerão no ecrã de administração.

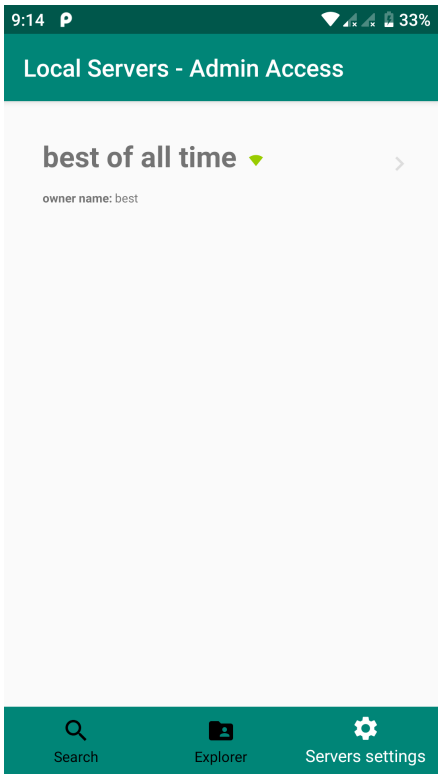


Figura 26: Ecrã de Lista de Servidores Privados com Acesso de Administração

Na versão atual apenas tem a capacidade de alterar as palavras-chaves de acesso ao conteúdo devendo, para tal, inserir a *password* atual de administração. Para além disto, existe a possibilidade de visualizar quais os utilizadores que têm acesso ao conteúdo do servidor privado.

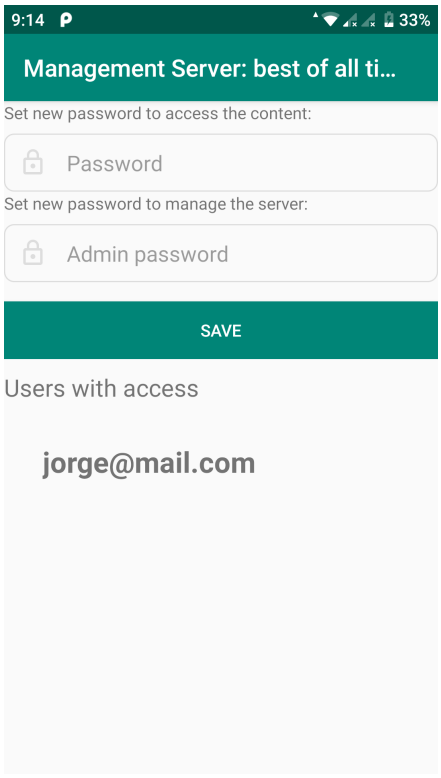


Figura 27: Ecrã de Administração do Servidor Privado

Quando as *passwords* de acesso são alteradas todos os utilizadores que possuiam uma autenticação válida para o servidor privado deixam de ter acesso ao mesmo e para voltar a explorar e consumir o conteúdo têm de efetuar uma autenticação válida inserindo as novas credenciais.

Por fim, caso utilizador pretenda encerrar a sua sessão, quer por opção própria quer seja para se autenticar com a conta de um outro utilizador, deve de pressionar o menu hambúrguer situado no topo esquerdo do ecrã e depois selecionar a opção "Terminar Sessão". Desta forma, a aplicação vai se comportar como se fosse a primeira vez que o utilizador estivesse a iniciar a aplicação e vai pedir novamente para efetuar uma autenticação ou então um novo registo no sistema.

B

TESTES

Para determinar o valor ótimo de similaridade do álbum e artista foi descarregada informação, a partir da API pública do Spotify, de alguns artistas, incluindo os seus álbuns e músicas associadas aos mesmos. Estes dados foram carregados na base de dados do Music Service e foram executados testes sobre os mesmos.

No caso da similaridade das músicas, como a equação é diferente, foi utilizado um outro conjunto de testes, criado manualmente, que permitiu achar o valor ótimo de similaridade entre as músicas.

B.1 CONJUNTO DE TESTES PARA A SIMILARIDADE ARTISTA E ÁLBUM

Foi criado um conjunto de dados, a partir da informação descarregada do Spotify e que foi carregada no sistema (será apresentada na secção B.3), que permitiu encontrar o valor ótimo de similaridade para o artista e álbum. Cada objeto de dados é constituído por outros dois objetos:

DATA: incluí os dados a validar (simulação dos dados a ser enviados pelo Local Agent), como o artista, álbuns associados e as músicas;

EXPECTED: incluí o resultado a ser esperado pelo algoritmo após validação dos dados de *input*. Indica se é um artista novo, se o artista já existe, mas com nome distinto ou se é o mesmo artista (mesma lógica para o álbum). Caso o resultado do algoritmo seja igual, então para este objeto de dados o algoritmo acertou, senão houve uma falha do mesmo.

```
1 [
2     {
3         data: {
4             artists: "carolina deslandes",
5             albumToInsert: "casa",
6             otherAlbuns: ["lado a lado"],
```

```
7             musics: [{ name: "adeus amor adeus", duration: 252093 }, { name: "
                  aleluia", duration: 278180 }, { name: "nuvem", duration: 313744
                   }]
8         },
9         expected: {
10            isNewArtist: false,
11            isSameArtistWithOtherName: false,
12            isSameArtist: true,
13            isNewAlbum: false,
14            isSameAlbumWithOtherName: false,
15            isSameAlbum: true
16        }
17    },
18    {
19        data: {
20            artists: "carol deslands",
21            albumToInsert: "kasa",
22            otherAlbuns: ["lado lado"],
23            musics: [{ name: "adeus amor adeus", duration: 242093 }, { name: "
                  alelluia", duration: 278180 }, { name: "nuvem", duration: 31374
                  4 }]
24        },
25        expected: {
26            isNewArtist: false,
27            isSameArtistWithOtherName: true,
28            isSameArtist: false,
29            isNewAlbum: false,
30            isSameAlbumWithOtherName: true,
31            isSameAlbum: false
32        }
33    },
34    ...
35 ]
```

O conjunto completo de dados pode ser consultado em `https://github.com/Jorgeoliv/PrivateServerStreamingService/blob/master/musicService/webServer/testData/artistAlbum.json`.

B.2 CONJUNTO DE TESTES PARA A SIMILARIDADE DA MÚSICA

Para testar o valor ótimo de similaridade da música foi criado um conjunto de dados de teste. Cada objeto de dados é constituído por três objetos:

MUSICA: E MUSICB as músicas a comparar, consituídas por nome e duração;

EXPECTED: o resultado expectável, *true* se as músicas foram semelhantes (as caraterísticas não precisam de ser exatamente iguais), *false* senão forem.

```
[
    {
        musicA: {
            name: 'adeus amor adeus', duration: '100000'
        },
        musicB: {
            name: 'adeus amor adeus', duration: '101000'
        },
        expected: true,
    },
    {
        musicA: {
            name: 'adeus amor adeus', duration: '100000'
        },
        musicB: {
            name: 'adues amor adues', duration: '101000'
        },
        expected: true,
    },
    ...
]
```

O conjunto completo de dados pode ser consultado em `https://github.com/Jorgeoliv/PrivateServerStreamingService/blob/master/musicService/webServer/testData/music.json`.

B.3 Conjunto de Dados Carregados no Sistema

Como referido anteriormente foi descarregado um conjunto de dados a partir da API do Spotify, para popular o sistema e poder realizar testes sobre o mesmo. Foram extraídos no total 100 músicas, 79 álbuns e 55 artistas.


```
{
   "artistas":[
      {
         "nome":"adele",
         "albuns":[
            {
               "nome":"21",
               "musicas":[
                  {
                     "nome":"rolling in the deep",
                     "duracao":228093
                  }
               ]
            }
         ]
      },
      {
         "nome":"marshmello",
         "albuns":[
            {
               "nome":"silence",
               "musicas":[
                  {
                     "nome":"silence",
                     "duracao":180822
                  }
               ]
            },
            {
               "nome":"alone",
               "musicas":[
                  {
                     "nome":"alone",
```

```
                    "duracao":273802
                }
            ]
        },
        {
            "nome":"happier",
            "musicas":[
                {
                    "nome":"happier",
                    "duracao":214289
                }
            ]
        }
    ]
  },
...
}
```

O conjunto completo de dados pode ser consultado em `https://github.com/Jorgeoliv/PrivateServerStreamingService/blob/master/musicService/webServer/testData/dataset.json`.