UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Gustavo Ferreira Custódio

# O software livre como ferramenta para distribuição de música de artistas independentes: Uma análise sobre o Fediverso e o Funkwhale

São João Del Rei

2025

UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI

Gustavo Ferreira Custódio

# O software livre como ferramenta para distribuição de música de artistas independentes: Uma análise sobre o Fediverso e o Funkwhale

Monografia apresentada como requisito da disciplina de Projeto Orientado em Compu- tação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Orientador: Flávio Luiz Schiavoni

Coorientador: Gabriel Rodrigues Chaves Carneiro

Universidade Federal De São João Del-Rei

Bacharelado em Ciência da Computação

São João Del Rei

2025

Gustavo Ferreira Custódio

# O software livre como ferramenta para distribuição de música de artistas independentes: Uma análise sobre o Fediverso e o Funkwhale

Monografia apresentada como requisito da disciplina de Projeto Orientado em Compu- tação II do Curso de Bacharelado em Ciência da Computação da UFSJ.

Trabalho aprovado. São João Del Rei, 28 de fevereiro de 2025:

---

**Dr. Flávio Luiz Schiavoni**
Orientador

---

**Gabriel Rodrigues Chaves Carneiro**
Orientador

---

**Dr. Acir Moreno Soares Júnior**
Convidado 1

---

**Dr. Guilherme de Castro Pena**
Convidado 2

São João Del Rei
2025

# Agradecimentos

# Resumo

O crescimento das redes sociais e plataformas baseadas em softwares proprietários trouxe preocupações sobre privacidade, controle de dados e a dependência de grandes corporações. Como alternativa a esse modelo, o Fediverso surge como um conjunto de plataformas descentralizadas que utilizam protocolos federados para permitir a comunicação entre diferentes servidores de maneira aberta e independente. Este trabalho investiga o papel do software livre no suporte a artistas independentes, com foco na plataforma Funkwhale, um serviço de *streaming* de áudio descentralizado e de código aberto. A pesquisa explora a arquitetura do Fediverso, o protocolo ActivityPub e o funcionamento do Funkwhale, além de apresentar a implementação de uma instância própria dessa plataforma. Os resultados mostram que o Funkwhale oferece uma alternativa viável ao modelo tradicional de distribuição musical, permitindo maior autonomia aos artistas e promovendo uma cultura digital mais aberta e acessível.

**Palavras-chaves**: software livre, Fediverso, Funkwhale, descentralização, streaming de áudio.

# Abstract

The growth of social networks and platforms based on proprietary software has raised concerns about privacy, data control, and dependence on large corporations. As an alternative to this model, the Fediverse emerges as a set of decentralized platforms that use federated protocols to enable communication between different servers in an open and independent manner. This work investigates the role of free software in supporting independent artists, focusing on Funkwhale, a decentralized and open-source audio streaming service. The research explores the architecture of the Fediverse, the ActivityPub protocol, and the operation of Funkwhale, while also presenting the implementation of a self-hosted instance of the platform. The results show that Funkwhale offers a viable alternative to the traditional music distribution model, allowing greater autonomy for artists and promoting a more open and accessible digital culture.

**Key-words**: open-source, Fediverse, Funkwhale, decentralization, audio streaming

# Lista de ilustrações

# Lista de abreviaturas e siglas

DAW - *Digital Audio Workstation*

URI - *Uniform Resource Identifier*

CNC - Conferência Nacional de Cultura

AWS - *Amazon Web Services*

TLS - *Transport Layer Security*

# Sumário

# 1 Introdução

Nos últimos anos houve uma explosão enorme na quantidade de pessoas navegando na internet e, consequentemente, interagindo e trocando informações. Atualmente, a lista das maiores empresas do mundo é dominada, por empresas do ramo de tecnologia, principalmente as com foco em desenvolvimento de sistemas e aplicativos como serviço, como a *Meta* e a *Google*. O grande aumento e popularidade dessas *Big Techs* fez com que seus líderes buscassem uma estratégia com o foco no marketing e no entendimento sobre os usuários da plataforma, o que é utilizado em sua grande maioria para a recomendação de conteúdo personalizado e propagandas (CAVA; GRECO; TAGARELLI, 2021).

Atualmente, a privacidade dos usuários nas redes sociais é um ponto crítico onde a grande parte dos usuários não se importa com a privacidade dos dados coletados por empresas ou não possui conhecimento dos riscos em relação à sua privacidade (CERRUTO et al., 2022). Com a crescente coleta e análise de dados pessoais por grandes plataformas, os usuários frequentemente se encontram em uma posição vulnerável, sem controle total sobre como suas informações são armazenadas, utilizadas ou compartilhadas. Os usuários são constantemente vigiados por essas plataformas e por meio de algoritmos de análise de sentimentos, os sistemas promovem recomendações de produtos, pessoas para seguir e lugares para se visitar, o que gera uma falha de privacidade grave, pois nenhum indivíduo aceita conscientemente ser monitorado de maneira tão intrusiva (RAO; KRISHNA; KUMAR, 2018).

Essa centralização de informações nas mãos de poucas corporações gera preocupações significativas. Entre elas, destacam-se questões relacionadas à privacidade, manipulação de informações, criação de bolhas algorítmicas e o impacto no comportamento social. O controle quase absoluto sobre os dados dos usuários permite que essas empresas exerçam uma influência profunda, não apenas sobre decisões individuais, mas também sobre dinâmicas sociais e políticas de nações inteiras (BADAWY; FERRARA; LERMAN, 2018; OTT, 2017), muitas vezes sem transparência ou regulação adequada, ampliando os riscos para a privacidade, a democracia e a autonomia dos cidadãos.

Além da questão da privacidade dos usuários, redes sociais controladas por empresas de algoritmo fechado são capazes de controlar a publicação e divulgação de conteúdo dentro da plataforma, atuando como um curador de conteúdo, sendo capaz de definir qual conteúdo será mais acessado e por consequência terá chance maior de viralizar e também aquele que não será tão divulgado e por consequência pouco acessado. Por um lado esse controle dos algoritmos sobre o conteúdo pode ser interpretado como uma estratégia de marketing, por outro existe o entendimento que isso também pode ser utilizado com uma

ferramenta de censura e manipulação de pensamentos e ideais.

Diante das crescentes preocupações com a violação da privacidade dos usuários na web e a ausência de uma regulamentação clara e eficaz, a União Europeia aprovou, no final de 2022, o *Digital Services Act* (DSA), ou Lei de Serviços Digitais. Esse marco regulatório tem como objetivo estabelecer regras mais rigorosas para os serviços digitais, promovendo maior transparência, segurança e proteção aos direitos dos usuários, além de responsabilizar as plataformas digitais por suas práticas (CAUFFMAN; GOANTA, 2021). Esse marco regulatório não apenas evidencia a necessidade urgente de conter o poder desproporcional das *Big Techs*, mas também reforça a centralidade da proteção de dados e da moderação de conteúdo nas discussões globais sobre o futuro da internet.

Em busca de alternativas para substituir as redes sociais centradas em grandes empresas, diversos projetos foram introduzidos com o objetivo de retirar esse domínio das grandes corporações e transferi-lo para os usuários. Esses projetos se destacam por suas características que os torna distintos das redes sociais corporativas, seja por sua ética, sua estrutura organizacional, suas tecnologias subjacentes, seus recursos ou o acesso ao código-fonte (MANSOUX; ABBING, 2020). Para garantir a descentralização da informação e permitir a troca de dados entre as diferentes instâncias de redes sociais não-centralizadas, surge um conceito de serviços federados e de federação de plataformas. O conjunto dessas plataformas é conhecido como Fediverso.

No fediverso os servidores são chamados de instâncias e cada instância pode definir de forma independente o seus termos de uso, políticas de moderação, localização do servidor e a interface gráfica, permitindo que os usuários possuam mais escolha (RIEMANN, 2022). Este ambiente virtual consiste em uma grande rede onde instâncias dessas plataformas existem de forma independente, mas que podem se comunicar entre si a partir de um mesmo protocolo. Dentre os serviços presentes no Fediverso atualmente, podemos destacar alguns, como o *Mastodon* e o *Friendica* como alternativas para o *X* (antigo Twitter), *PeerTube*, como alternativa para o Youtube e o caso de estudo do presente trabalho, *Funkwhale*, que se trata de uma plataforma de *streaming* de áudio que serve como alternativa para o *Spotify* e outras. A Seção 2.1 traz mais informações sobre a organização e comunicação dos Fediversos.

O fediverso vem chamando cada vez mais atenção e vem sendo cada vez mais adotado ao redor do mundo, a *Next Generation Internet* (NGI) [1] possui um projeto chamado *NGI Fediversity* que busca investir no fediverso em busca de criar um espaço prático, seguro e amigável ao usuário. Além disso, a União Europeia possui um servidor independente na rede social Mastodon, onde estão inseridos os perfis relacionados a mesma, como

[1] A *Next Generation Internet* (NGI) é uma iniciativa da Comissão Europeia (EC) que busca moldar o desenvolvimento e a evolução da Internet em uma Internet de Confiança, uma Internet que atenda às necessidades fundamentais das pessoas, incluindo confiança, segurança e inclusão, ao mesmo tempo em que reflita os valores e as normas que todos os cidadãos desfrutam na Europa.

o perfil da Comissão Europeia e o da NGI. O fediverso e sua arquitetura serão abordados mais profundamente na sessão 2.1.

Com a popularização do Fediverso, novas plataformas descentralizadas foram desenvolvidas para competir com plataformas centradas em grandes empresas, como Youtube e Spotify, por exemplo. Uma que ganhou destaque com o passar dos anos foi a plataforma de *streaming* de áudio chamada Funkwhale. Esse serviço de *streaming* surgiu como uma alternativa ao Spotify, porém descentralizada e de código aberto. O Spotify se trata de uma plataforma de *streaming* de áudio que ganhou bastante popularidade nos anos recentes, assim como as principais empresas do ramo de tecnologia, tem investido bastante em algoritmos de recomendação para sugerir músicas com base no gosto do usuário. Esses algoritmos priorizam músicas que estão sendo mais ouvidas, como a criação de playlists automáticas na plataforma com base na quantidade de vezes que foi tocada (ANDERSON et al., 2020). Além disso, o Spotify exige que artistas que desejam se cadastrar na plataforma escolham uma distribuidora de música para gerenciar as mesmas dentro do Spotify, de forma que as músicas só podem ser publicadas no serviço de *streaming* pela distribuidora. O Funkwhale apresenta uma alternativa para artistas, promovendo mais visibilidade para artistas pequenos e uma interface simples, facilitando o processo de upload de músicas sem precisar de distribuidoras digitais como no Spotify. Na Seção 2.2 trazemos mais sobre o funcionamento desta plataforma.

A utilização de softwares livres é uma grande ferramenta para auxiliar artistas independentes e a cultura *underground*. A palavra *underground* vem do inglês e seu significado literal é subterrâneo. No âmbito da música, *underground* é algo que foge dos padrões comerciais da indústria fonográfica (SOUSA; SCHIAVONI, 2023). A produção independente se torna um desafio visto que as principais ferramentas necessárias para produção e distribuição de músicas apresentam um custo elevado para artistas que estão começando e ainda não possuem uma grande visibilidade. Com o tempo surgiram ferramentas de código aberto que se apresentam como alternativas gratuitas para as soluções pagas consolidadas no mercado, como, por exemplo, o LMMS e o Ardour, softwares de produção musical livre e de código aberto. A união de softwares de código aberto com o movimento *underground* é algo bastante discutido na literatura e será abordado com mais detalhes no capítulo 3.

O presente projeto visa entender um pouco mais sobre o funcionamento e a arquitetura principal do fediverso a partir de um estudo de caso sobre a plataforma de streaming de áudio de código aberto *Funkwhale*. Os principais objetivos são entender o impacto de sistemas federados de código aberto no movimento *underground*, entender a arquitetura do fediverso e principalmente do *Funkwhale* e analisar como os protocolos federados são utilizados na plataforma, buscando destrinchar o processo de implementar uma instância própria desse sistema no capítulo 4 para a distribuição de conteúdo musical

de forma gratuita e independente.

Para isso, realizamos a instalação e configuração de uma instância da plataforma Funkwhale utilizando o *docker* para executar os serviços necessários para o funcionamento da instância e a ferramenta de criação de servidores virtuais *AWS Ligthsail* para publicar a aplicação na web. Este processo foi apresentado na seção 4 de forma mais detalhada apresentando os resultados na seção 5, como a interface da aplicação, as principais ferramentas disponíveis para os usuários e a maneira como funciona o processo de federação com outras instâncias da plataforma.

# 2 Referencial Teórico

Este capítulo apresentará definições dos conceitos mais importantes deste trabalho, o Fediverso, seus principais protocolos e a plataforma Funkwhale.

## 2.1 Fediverso

Fediverso origina-se da união das palavras Universo e Federação e se trata de uma rede de sistemas interativos que comunicam entre si a partir de um mesmo protocolo de rede (MANSOUX; ABBING, 2020). A ideia principal do Fediverso é permitir que usuários de diversas comunidades e/ou plataformas diferentes possam interagir sem necessariamente estarem na mesma plataforma, permitindo o poder de escolha dos usuários em relação a qual plataforma o mesmo deseja utilizar (MANSOUX; ABBING, 2020). O Fediverso funciona com base no protocolo *ActivityPub*, que implementa uma camada de comunicação entre cliente-servidor e outra entre servidor-servidor. Detalhes deste protocolo apresentados na Seção 2.1.1. A Figura 1 apresenta uma representação visual no Fediverso, onde cada logo representa uma instância específica de uma plataforma e as setas representam a intercomunicação dos sistemas.

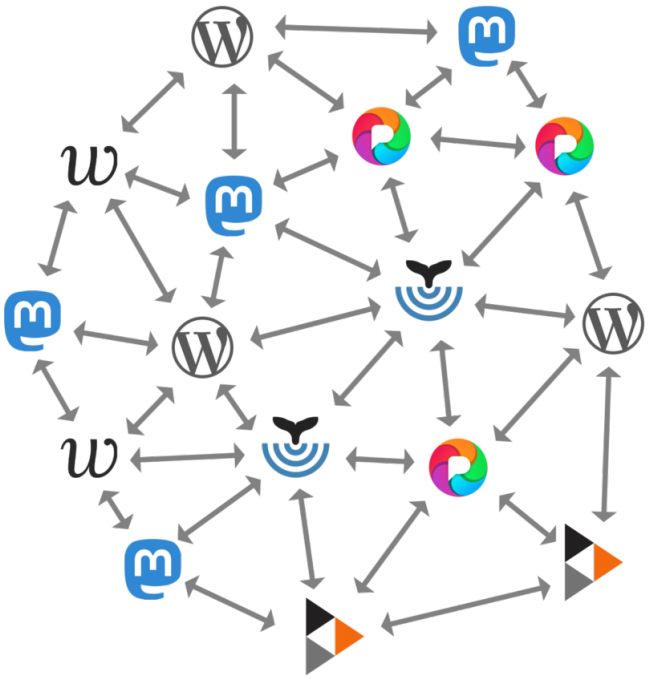

Figura 1 – Representação visual do Fediverso

A origem do conceito do Fediverso é desconhecida, porém acredita-se que surgiu

no inícios dos anos 2000s a partir de uma abstração dos protocolos utilizados para envio de emails. A primeira manifestação do Fediverso ocorreu em meados de 2008 quando o desenvolvedor norte-americano Evan Prodromou lançou a plataforma *Identi.ca*. Essa aplicação foi desenvolvida como uma plataforma de microblogging gratuita e de código aberto, alternativa para a plataforma de microblogging mais utilizada na época, o *Twitter*. A *Identi.ca* funcionava pelo protocolo OpenMicroBlogging (OMB) também desenvolvido por Evan Prodromou especificamente para essa plataforma. O OMB funcionava como um serviço de assinatura, onde um usuário de uma plataforma A, poderia autorizar o envio de notificações de um outro usuário de uma plataforma B para a sua "caixa de entrada"(PRODROMOU, 2008).

Com o passar dos anos, o conceito de fediverso amadureceu, os protocolos evoluíram e novas plataformas surgiam, como *Mastodon*, *diaspora*, *Funkwhale*, *Peertube*, etc. Segundo informações do site *Fediverse Party* [1], um site que reúne informações das instâncias públicas das principais plataformas federadas, o Fediverso possui atualmente cerca de 13 milhoẽs e 700 mil contas registradas, com 2 milhões de usuários ativos e 17 mil instâncias únicas.

Atualmente existem cerca de 12,895 instâncias de diferentes aplicações federadas segundo o site *To the Fediverse*. A principal aplicação federada existente nos dias de hoje é a rede social *Mastodon*, que possui um funcionamento bastante semelhante ao *X* (antigo Twitter), mas além dessa plataforma existem diversas outras análogas a outros serviços de código fechado. Dentre todas as aplicações federadas disponíveis atualmente, podemos destacar as seguintes:

- **Mastodon**: Uma rede federada e de código aberto para *Microblogging* similar ao X.
- **Funkwhale**: Uma plataforma federada e de código aberto para *streaming* de músicas.
- **Pixelfed**: Uma rede social de código aberto que permite o compartilhamento de imagens, similar ao Instagram.
- **Peertube**: Aplicação *open source* para envio e compartilhamento de vídeos similar ao Youtube.
- **Plume**: Uma aplicação de *microblogging* que permite que usuários monte um blog pessoal.
- **Hubzilla**: Uma aplicação de código aberto que permite o envio e o compartilhamento de arquivos, similar ao Google Drive.

[1] https://fediverse.party/en/

Podemos traçar paralelos entre algumas aplicações do Fediverso com plataformas de software proprietário. A tabela 1 apresenta softwares consolidados de código fechado e suas alternativas federadas e de código aberto:

| **Tipo de aplicação** | **Aplicações com softwares proprietários** | **Aplicações federadas** |
|---|---|---|
| Rede social de textos curtos | Twitter | Mastodon/Friendica |
| *Streaming* de músicas e podcasts | Spotify | Funkwhale |
| Compartilhamento de vídeos | YouTube | PeerTube |
| Compartilhamento de fotos | Instagram | PixelFed |
| *Microblogging* | Blogger | Plume/WriteFreely |
| Compartilhamento de arquivos | Google Drive | Hubzilla |

Tabela 1 – Comparação entre softwares proprietários e suas versões federadas

Grande parte das principais redes sociais existentes atualmente não permitem federação dentro de sua infraestrutura. Aplicações de troca de mensagens de texto como WhatsApp e Telegram utilizam o protocolo XMPP (*Extensible Messaging and Presence Protocol*), que é composto por um conjunto de tecnologias de código aberto para mensagens instantâneas, bate-papo em grupo, chamadas de voz e vídeo e distribuição de conteúdo. O protocolo é baseado no formato XML, que permite organizar documentos de forma estruturada e hierárquica, parecido com o HTML. Este protocolo permite a comunicação entre servidor-cliente, cliente-cliente e servidor-servidor. Basta realizar uma adaptação deste protocolo para que o mesmo possa ser utilizado para federar essas plataformas, permitindo a comunicação entre servidores de aplicações diferentes. Isso indica que a ausência de federação nessas plataformas não é uma limitação técnica, mas sim uma escolha comercial das empresas que as desenvolvem.

Uma exceção recente a essa tendência de outras redes sociais é o Threads, plataforma da Meta, que anunciou suporte à federação por meio do protocolo ActivityPub, o mesmo utilizado pelo Mastodon e outras redes do Fediverso. Esse movimento indica uma possível mudança na abordagem das grandes empresas em relação à inter-comunicação entre plataformas. Essa alteração na maneira como o Threads opera pode indicar um movimento que está surgindo no qual as *Big Techs* estão considerando a possibilidade de se federar algum dia. Por mais que este movimento tenha aumentado a atenção em relação ao fediverso e a aplicações federadas, ainda não se sabe até que ponto a integração do Threads com o fediverso será completa ou se haverá restrições impostas para manter o controle sobre a base de usuários.

Em abril de 2022, a Comissão Europeia anunciou na rede social X que estava ingressando na plataforma federada e de código aberto Mastodon. A decisão aconteceu em um momento onde a preocupação sobre como a plataforma se preocupação com a privacidade do usuário era uma grande pauta de debate após a aquisição da plataforma pelo empresário sul-africano Elon Musk. Atualmente a Comissão Europeia possui uma instân-

cia própria na plataforma Mastodon [2], o que permite que a maneira como os dados são armazenados e tratados está completamente no controle da própria comissão, garantindo a segurança e a privacidade dos usuários desta instância.

As redes sociais federadas oferecem uma alternativa viável para lidar com questões de privacidade, controle de dados e dependência de softwares proprietários de grandes corporações. Ao permitir que qualquer pessoa ou organização hospede sua própria instância, estas plataformas descentralizadas garantem maior autonomia sobre a gestão dos dados e aplicação de políticas de moderação. Diferente das redes sociais centralizadas, onde as regras são impostas por uma única empresa, o modelo federado distribui o poder entre diversas instâncias, possibilitando um ambiente mais transparente e alinhado aos interesses dos usuários. Além disso, a federação promove a comunicação entre instâncias de diferentes plataformas, reduzindo a concentração de informações sensíveis em servidores únicos e minimizando os riscos de censura, exploração comercial excessiva e vazamento de dados. A seção 2.1.1 irá apresentar o protocolo que forma a infraestrutura principal para a federação de diferentes plataformas.

### 2.1.1 ActivityPub

O ActivityPub é o protocolo utilizado no Fediverso para realizar a troca de informações entre as instâncias do mesmo. Esse protocolo é uma recomendação do *World Wide Web Consortium* (W3C), que se trata de uma das principais organizações que define padrões e diretrizes para guiar a construção de uma Web baseada nos princípios de acessibilidade, internacionalização , privacidade e segurança [3].

Este protocolo foi lançado em 28 de janeiro de 2018 e se trata de uma evolução/adaptação do protocolo *OStatus* e da api do *Pump.io*. O *OStatus* é um protocolo também desenvolvido por Evan Prodromou e é utilizado até os dias atuais pela *Friendi.ca* e se trata de um protocolo onde cada sistema que o utiliza produz um feed *Atom*, que consiste em um documento no formato XML (linguagem de marcação) que descreve uma lista de informações correlatas, chamadas de feeds, cada feed possui um conjunto de itens, chamados de entradas, onde cada entrada possui um conjunto de metadados (título e descrição por exemplo) [4]. Para enviar informações do feed para os usuários, é utilizado o *PubSubHubbub*, que consiste em uma aplicação do modelo de assinatura *Push Subscription* onde os usuários se inscrevem para receber informações de um feed e essas informações são enviadas para os usuários inscritos no feed. A ideia principal o *OStatus* é implementar feeds *Atom* e enviar essas informações para outros sites utilizando o *Push Subscription*.

[2] https://ec.social-network.europa.eu/explore
[3] https://www.w3.org/
[4] https://www.rfc-editor.org/rfc/rfc4287.html

O *Pump.io* se trata de um servidor de transmissão de dados textuais para redes sociais. O Pump utiliza um formato chamado *ActivityStreams 1.0* [5] para a formatação dos dados e cada conta do *Pump.io* possui dois feeds, um de entrada de dados, chamado de *Activity Inbox*, onde o usuário pode ler os post enviados para ele e um de saída de dados, chamado de *Activity Outbox*, onde o usuário posta suas atividades para outros usuários, cada feed se trata de um conjunto de atividades [6]. De forma simples, uma atividade consiste de um ator (pessoa que realiza a atividade), um verbo (indica a atividade que está sendo realizada), um objeto e um alvo. Uma atividade expressa uma ação que um determinado ator realiza com ou no objeto em questão (Lucas postou uma foto no seu álbum, por exemplo). Uma *Activity Stream* se trata de uma coleção de atividades no formato JSON. A figura 2 apresenta a representação de uma atividade no formato *ActivityStreams 1.0*, onde o usuário Martin Smith realiza uma postagem no seu blog pessoal (Martin's Blog).

```
{
  "published": "2011-02-10T15:04:55Z",
  "actor": {
    "url": "http://example.org/martin",
    "objectType" : "person",
    "id": "tag:example.org,2011:martin",
    "image": {
      "url": "http://example.org/martin/image",
      "width": 250,
      "height": 250
    },
    "displayName": "Martin Smith"
  },
  "verb": "post",
  "object" : {
    "url": "http://example.org/blog/2011/02/entry",
    "id": "tag:example.org,2011:abc123/xyz"
  },
  "target" : {
    "url": "http://example.org/blog/",
    "objectType": "blog",
    "id": "tag:example.org,2011:abc123",
    "displayName": "Martin's Blog"
  }
}
```

Figura 2 – Exemplo de atividade no formato *ActivityStreams 1.0*

O protocolo ActivityPub se trata de uma evolução do *OStatus* e *Pump.io*, na qual utiliza do padrão *ActivityStreams 2.0*, para processamento dos dados (uma evolução do protocolo utilizado no *Pump.io*) e também possui o conceito de caixa de entrada e de saída (*Inbox* e *Outbox*). Este protocolo apresenta duas camadas de rede, uma que lida com a comunicação entre servidores (onde plataformas descentralizadas podem compatilhar

[5] https://activitystrea.ms/specs/json/1.0/
[6] https://github.com/pump-io/pump.io/blob/master/API.md

informações e outra que lida com a comunicação cliente-servidor [7]. Aplicações federadas podem implementar somente uma das camadas ou as duas. Neste protocolo, os usuários são representados por "atores", onde cada ator possui uma caixa de entrada, onde eles recebem mensagens do mundo, e uma caixa de saída, onde eles compartilham mensagens com outros usuários de outros servidores, como pode ser visualizado na figura 3.

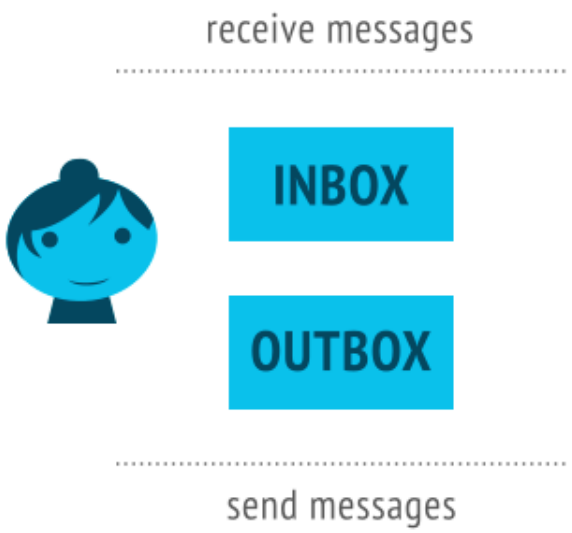


Figura 3 – Atores no protocolo *ActivityPub*

As caixas de entrada e saída são fundamentais para estabelecer a comunicação entre usuários de diferentes servidores. Tanto a caixa de entrada quanto a de saída realizam requisições GET e POST dependendo do contexto de cada requisição. Atores podem realizar requisições POST diretamente para a caixa de entrada de outros atores (comunicação servidor-servidor), requisições GET para sua caixa de entrada para ler suas mensagens (comunicação cliente-servidor), requisições POST para a sua caixa de saída para serem enviadas para o mundo (cliente-servidor) e requisições GET para a caixa de saída de outro ator (cliente-servidor e/ou servidor-servidor). A figura 4 apresenta uma representação visual da comunicação entre atores e suas caixas de entrada e saída.

[7] https://www.w3.org/TR/activitypub/

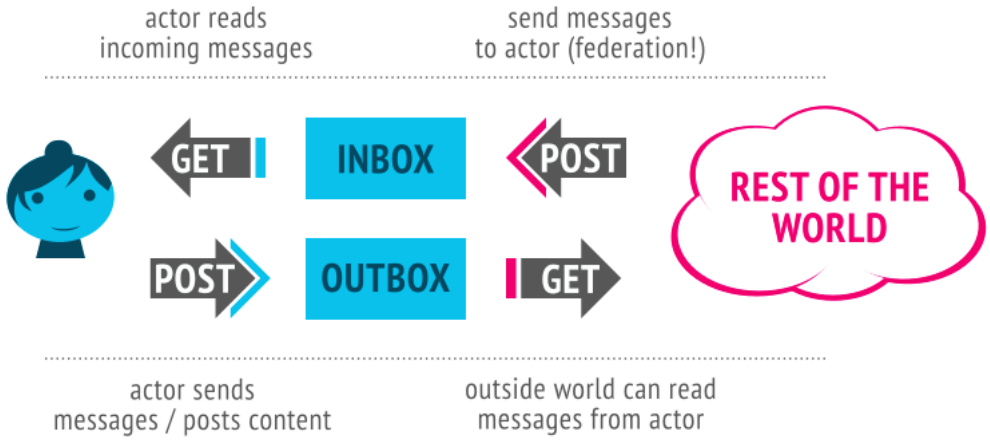


Figura 4 – Representação de requisições na caixa de entrada e saída

Como mencionado o protocolo *ActivityPub* utiliza do formato *ActivityStreams 2.0* em suas requisições, esse formato é uma evolução da versão 1.0, que possuía algumas limitações. As principais evoluções apresentadas da segunda versão em relação a primeira foram a unificação do parâmetro *verb* (que indica a ação realizada) e do *objectType* (que indica o tipo do objeto que sofreu a ação) em um único parâmetro *type* (que indica o tipo de ação realizada), a introdução do tipo *Link* que serve para fornecer uma descrição rica de links e a compatibilidade com o JSON-LD. JSON-LD é um formato de arquivos JSON, que possui a premissa de ser um formato leve e fácil para a leitura e escrita por humanos [8]. A ideia de adicionar compatibilidade do JSON-LD ao formato *ActivityStreams 2.0* é sustentada pela capacidade de documentos do tipo JSON-LD fornecerem opções para lidar com dados encadeados (LinkedData), permitindo a extensão do vocabulário do *ActivityStreams*. JSON-LD utiliza uma propriedade especial chamada de *context*, essa propriedade, como o nome indica, fornece o contexto da atividade, permitindo o uso de termos de atalho para uma comunicação mais eficiente sem perder a precisão [9]. Basicamente termos são parâmetros que podem ser usados no formato JSON-LD e o contexto é o responsável por determinar esses termos. A imagem 5 apresenta um exemplo de requisição no formato *ActivityStreams 2.0*, onde o usuário Martin adicionou um artigo ao seu blog. Como é possível observar, as tags *verb* e *objectType* presentes na figura 2 (exemplo de atividade no protocolo *ActivityStreams*) foram alteradas para o parâmetro type, que exemplifica o tipo de ação ou objeto da atividade. Além disso é possível observar a presença do contexto como um link para a página de termos do *ActivityStreams 2.0* indicando o "vocabulário"utilizado neste documento.

[8] https://json-ld.org/
[9] https://www.w3.org/TR/json-ld/#the-context

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "Martin added an article to his blog",
  "type": "Add",
  "published": "2015-02-10T15:04:55Z",
  "actor": {
   "type": "Person",
   "id": "http://www.test.example/martin",
   "name": "Martin Smith",
   "url": "http://example.org/martin",
   "image": {
     "type": "Link",
     "href": "http://example.org/martin/image.jpg",
     "mediaType": "image/jpeg"
   }
  },
  "object" : {
   "id": "http://www.test.example/blog/abc123/xyz",
   "type": "Article",
   "url": "http://example.org/blog/2011/02/entry",
   "name": "Why I love Activity Streams"
  },
  "target" : {
   "id": "http://example.org/blog/",
   "type": "OrderedCollection",
   "name": "Martin's Blog"
  }
}
```

Figura 5 – Exemplo de atividade no formato ActivityStreams 2.0

## 2.2 FunkWhale

O Funkwhale é uma plataforma gratuita de streaming de áudio descentralizada e de código aberto que começou a ser desenvolvida em 2015, projetada para que os usuários possam compartilhar e escutar músicas e podcasts de forma livre. Trata-se de um sistema auto-hospedado, configurado e executado em servidores controlados diretamente pelos próprios usuários, garantindo maior autonomia e privacidade. A plataforma utiliza o protocolo ActivityPub, permitindo a comunicação com outras plataformas federadas, bem como a interação entre diferentes instâncias do Funkwhale. Essa abordagem promove uma experiência integrada, descentralizada e colaborativa. Uma instância do Funkwhale é chamada de *pod*, cada *pod* se trata de um servidor independente, que possui seu próprio conteúdo e suas regras. Nos dias atuais este serviço de streaming possui cerca de 83 servidores ativos com cerca de 12mil contas registradas, sendo compatível com plataformas web e aplicativos para sistemas Android e IOS.

No Funkwhale, os canais são uma funcionalidade essencial que permite aos usuários

publicar e compartilhar seus próprios conteúdos de áudio dentro da plataforma e em toda o Fediverso. Eles são uma forma prática e poderosa de divulgar criações, como músicas ou podcasts, para um público global. Existem dois tipos de canal no Funkwhale, um para a publicação de podcasts e outro para a publicação de discografia de um artista. O conteúdo publicado como podcast pode ser acessado diretamente por outros usuários do Funkwhale ou por qualquer aplicativo de podcast que suporte RSS (chamados de podcatchers) além de também poderem ser importados para dentro da plataforma como feeds RSS. De forma simplificada, canais no Funkwhale seria como um perfil de um determinado artista ou podcast.

Outra funcionalidade essencial no Funkwhale são as bibliotecas, que são utilizadas para organizar e gerenciar o conteúdo de áudio enviado para o seu pod. Elas permitem que os usuários categorizem coleções de músicas ou outros arquivos de áudio e controlem quem pode acessá-los, oferecendo flexibilidade e privacidade na administração do conteúdo. Ao criar uma biblioteca no Funkwhale é possível definir diferentes visibilidades do conteúdo permitindo que você compartilhe suas músicas com muitas pessoas ou permita que apenas algumas pessoas tenham acesso. Atualmente, existem três tipos de visibilidade disponíveis para as bibliotecas no Funkwhale:

1. Visível para todos no Fediverso: O conteúdo da biblioteca pode ser acessado por qualquer pessoa que interaja com o seu pod, incluindo usuários de outros pods na rede federada.

2. Visível apenas para usuários do seu pod: Apenas os usuários registrados na sua instância (pod) poderão visualizar o conteúdo da biblioteca. Esta opção exclui usuários de outros pods.

3. Visível apenas para você: O conteúdo da biblioteca é restrito exclusivamente ao seu acesso, sem que outros usuários possam visualizá-lo.

Além dos canais e bibliotecas, o Funkwhale oferece outras ferramentas para organizar e gerenciar arquivos de áudio, semelhantes às encontradas em diversas plataformas de streaming. Os usuários podem criar filas de reprodução, permitindo que músicas sejam adicionadas para serem ouvidas em uma sequência personalizada. Também é possível montar playlists, organizando músicas com base em estilos, temas ou qualquer outro critério desejado. O Funkwhale conta ainda com a funcionalidade de rádios, que geram playlists automáticas baseadas em critérios específicos, como estilo musical ou artista. Além disso, os usuários podem criar uma lista de favoritos, uma playlist especial dedicada às músicas que mais gostam.

O Funkwhale ainda possui suporte a plugins que inserem ferramentas adicionais à plataforma. Os principais que estão disponíveis no website oficial da plataforma são:

1. Scrobbler: é um plugin que permite que você envie (scrobble) as músicas que ouviu para um serviço de scrobbling. Scrobbling significa enviar as músicas e ou podcasts que você ouve para sites externos para ter informações sobre o que o usuário escuta e recomendações personalizadas. Scrobble ajuda a criar um perfil dos seus gostos musicais e também permite que você mantenha um registro das músicas que ouviu.

2. ListenBrainz: permite que você envie (scrobble) as músicas que ouviu para a sua conta do ListenBrainz. ListenBrainz é uma plataforma de código aberto registra as músicas que você ouve e fornece insights sobre seus hábitos de escuta. Com o ListenBrainz, é possível acompanhar suas preferências musicais, descobrir novas músicas por meio de recomendações personalizadas e compartilhar seu gosto musical com outras pessoas.

3. Maloja: outro plugin que permite que você envie as músicas que ouviu para um servidor Maloja. Maloja também se trata de uma plataforma de código aberto que permite que o usuário armazene seu histórico de músicas ouvidas e acompanhe suas preferências musicais.

A plataforma ainda disponibiliza ferramentas para moderação de conteúdo e usuários e usuários dentro de uma instância. Os moderadores das instâncias são responsáveis por estabeler regras dentro da plataforma, podendo moderar usuários e domínios, bem como gerenciar o conteúdo de uma biblioteca. O sitema apresenta um opção para os usuários denunciarem outros usuários que violaram alguma regra da instância. Essa denúnica é enviada para um moderador que possui a permissão para desabilitar o login, alterar as permissões ou diminuir a quantidade de upload do usuário. Além disso a plataforma também apresenta um opção de validação de registro de usuários, que faz com que somente os usuários que tiverem seu cadastro aprovado poderão acessar a instância. Outras opções de moderação permitem com que um moderador gerencie o conteúdo do pod, excluíndo músicas ou alterando a visibilidade de bibliotecas, ou controle a comunicação da sua instância com outras instâncias do funkwhale e/ou de outras aplicações do fediverso.

A federação no Funkwhale permite que usuários acompanhem usuários e músicas de outras instâncias, permitindo que usuários visualizem e interajam com outros usuários e bibliotecas. O Funkwhale utiliza o protocolo *ActivityPub* para realizar a comunicação entre as instâncias e o formato *ActivityStreams2.0* para realizar essa comunicação. Além disso, o Funkwhale também utiliza o *WebFinger* [10], que consiste em um protocolo utilizado para buscar informações estáticas sobre pessoas ou outras entidades a partir de URIs (Identificador de Recursos Universal, que é um identificador de recursos na web, composto pelo URL mais o identificador específico do recurso), e é utilizado no Funkwhale para acessar informações sobre recursos de outras instâncias. O banco de dados do Funkwhale é

[10] https://datatracker.ietf.org/doc/html/rfc7033

modelado a partir das entidades do protocolo *ActivityPub* e cada usuário do sistema é um ator, que performa atividades dentro do mesmo. Quando um usuário realiza uma atividade, como carregar uma música para uma biblioteca, são disparados dois tipos de entrega de conteúdo, um que envia para os atores na mesma instância e outra que envia para os atores de instâncias remotas. O Funkwhale ainda possui um ator específico para cada instância chamado de ator de serviço, este fica responsável por distribuir as atividades e este possui uma autoridade, de forma que quando o mesmo envia uma atividade relacionada a um objeto no domínio, todos os domínios remotos são capazes de identificar a autenticidade do mesmo [11]. No momento de escrita deste trabalho a federação na plataforma ocorre a partir das bibliotecas de músicas, de forma que quando um usuário cria uma biblioteca é gerada uma URL com um identificador único da mesma, este link pode ser utilizado por outros usuários em outros domínios para acessar o conteúdo da biblioteca. Ao copiar o link da biblioteca é mostrado ao usuário algumas informações da mesma, como nome e quantidade de músicas e também aparece um botão que permite com que o usuário envie uma solicitação para o dono para seguir a mesma. Se aprovada a solicitação o conteúdo da biblioteca é exibido para o usuário dentro da plataforma e é exibida uma opção para escanear a mesma, esta funciona para atualizar esporadicamente a mesma, adicionando as músicas que foram incluídas e removendo as que foram retiradas após o último scan.

Em relação a arquitetura utilizada, o servidor do Funwhale é desenvolvido na linguagem Python utilizando o Django REST Framework para desenvolver a sua api [12]. Além disso, o Funkwhale utiliza o PostgreSQL como sistema de gerenciamento de banco de dados, o Redis para caching e otimização da comunicação entre a base de dados e o cliente, além do Celery, composto pelo Worker e pelo Beat Task Scheduler. O Celery Worker executa tarefas de forma assíncrona, permitindo que a API se concentre em respostas em tempo real, enquanto o Celery Beat gerencia tarefas recorrentes, controladas por um agendador (*scheduler*). O Celery é responsável pelas tarefas relacionadas com a parte de federação, buscando e enviando informações (atividades), entre as instâncias. As instâncias do Funkwhale podem ser plataformas web, aplicativos Android ou aplicativos baseados na api Subsonic, e utilizam de uma ferramenta de proxy reverso para realizar a comunicação com o servidor. A figura 6 representa a arquitetura geral de uma instância do funkwhale.

[11] https://docs.funkwhale.audio/developer/federation/

[12] O Django REST Framework é um pacote de ferramentas para desenvolver uma api web. Apis web são responsáveis por estabelecer a conexão entre o cliente (que nesse caso é a instância web ou o aplicativo do Funkwhale) e o banco de dados. https://www.django-rest-framework.org/

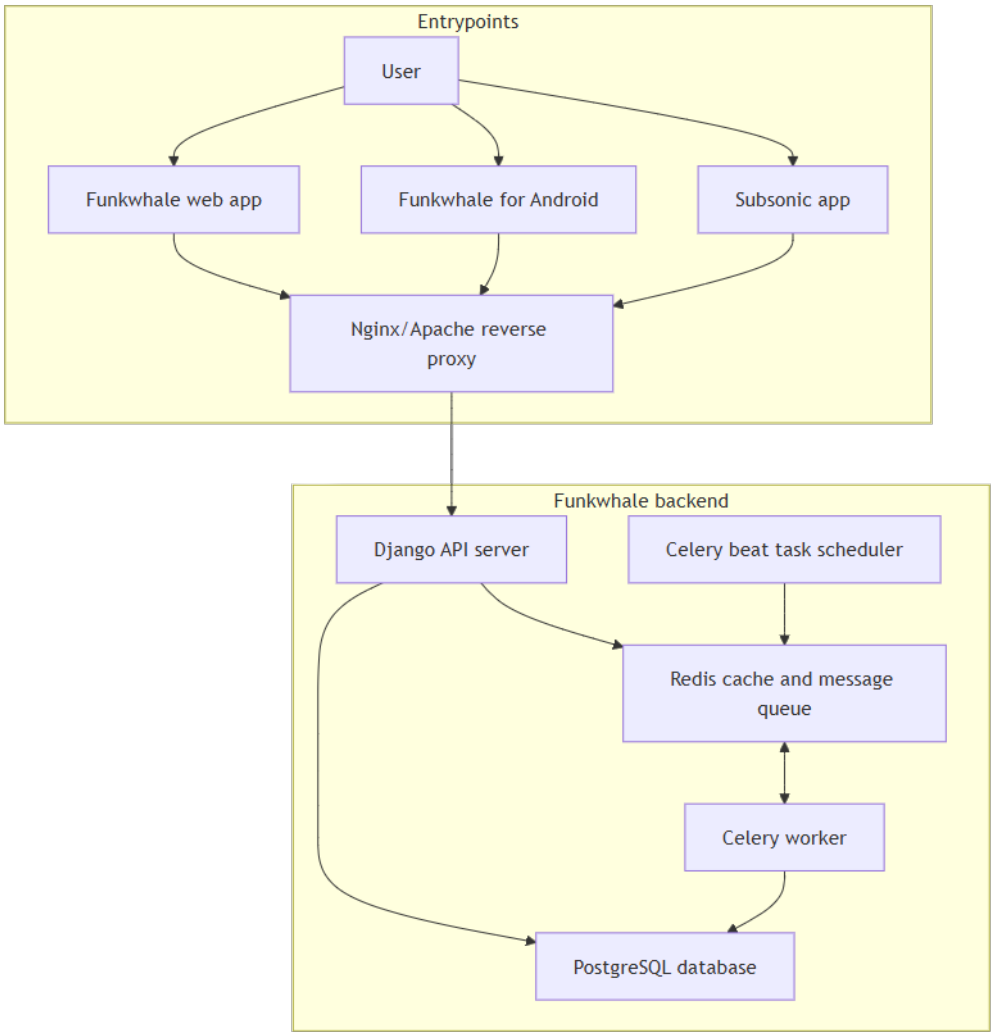


Figura 6 – Arquitetura do Funkwhale

# 3 Trabalhos Relacionados

A ideia de utilizar softwares livres como ferramenta para artistas independentes não é algo novo. Plataformas de código aberto se apresentam como ótimas alternativas para soluções pagas que cobram altos valores pela sua licença e distribuição. Artistas independentes do movimento underground não possuem uma grande visibilidade e com isso não geram receita suficiente para arcar com o valor de ferramentas pagas com o seu trabalho. Todos os trabalhos que serão apresentados nessa seção possuem algo em comum, o uso de software livre como ferramenta para produção cultural.

## 3.1 Depurando o Underground: artistas independentes capacitando-se em produção musical com software livre

Em (SOUSA; SCHIAVONI, 2023), o autor evidencia os principais desafios e dificuldades para artistas do meio *underground* produzirem suas músicas. O principal desafio encontrado por artistas independentes é o alto valor do investimento inicial feito para construir uma estrutura capaz de produzir composições musicais com qualidade. Softwares para produção de faixas instrumentais e mixagem e masterização (*DAW*) em grande maioria são proprietários e possuem uma licença paga. Essas *DAWs* também possuem diversos plugins que auxiliam bastante na produção, porém a grande maioria também é paga.

Isso gera uma problemática para artistas independentes do meio *underground* que em sua grande maioria são periféricos e com fragilidades econômicas. Isso se torna um grande desafio para estes produtores visto o custo elevado de licenças de softwares de produção de áudio e seus plugins e também visto que a pirataria infringe o Art. 184 da lei nº 2848, de 07 de dezembro de 1940 podendo gerar uma pena de detenção de três meses a um ano ou multa para aqueles que utilizarem softwares proprietários sem a sua devida licença.

Devido aos problemas evidenciados o autor apresenta uma alternativa gratuita e de código aberto para as *DAWs* proprietárias, o LMMS. O LMMS se trata de um software livre e de código aberto que apresenta uma interface simples e similar as soluções pagas consolidadas do mercado, como por exemplo o *FL Studio*. No projeto, o LMMS é utilizado em oficinas teórico-práticas de produção musical, para fomentar a produção musical independente. Essa solução gratuita apresenta uma interface simples e intuitiva para usuários inexperientes ao mesmo tempo que possui uma interface semelhante a outros softwares permitindo uma rápida adaptação para usuários já familiarizados com outras

ferramentas.

O uso do LMMS para a produção musical independente se provou algo eficiente para apresentar para apresentar o ambiente da produção fonográfica para produtores iniciantes, bem como fornecer uma alternativa sem custos e completa para produtores experientes do cenário *underground*.

## 3.2 O uso da tecnologia para práticas musicais em grupos de percussão

O trabalho "O uso da tecnologia para práticas musicais em grupos de percussão"(RIBEIRO et al., 2024), apresenta uma ferramenta alternativa para o ensino informal de música baseado em uma aplicação de celular. A proposta do projeto surgiu a partir da necessidade de auxiliar coletivos musicais, conhecidos como batucadas, nos estudos de percussão.

A partir dessa premissa foi desenvolvido um aplicativo para celular com o objetivo de ajudar os músicos na prática de percussão e ritmo. O aplicativo conta com uma interface intuitiva, onde a tela inicial oferece opções para ajustar o andamento e o volume geral, além de possibilitar o acionamento e desligamento da máquina rítmica. A segunda tela do sistema apresenta a opção de escolher os instrumentos que serão utilizados na composição, como exemplo elementos de uma bateria. Após selecionar os instrumentos que serão utilizados, o usuário é levado para uma tela onde ele pode montar de forma simples e intuitiva uma percussão.

O sistema foi desenvolvido a partir da ferramenta *Pure Data*, que se trata de uma linguagem de programação visual open source que permite o processamento e manipulação de áudio, vídeo, sensores e interfaces MIDI. A programação no *Pure Data* é feita a partir de blocos que se conectam entre si. Para transportar a lógica do *Pure Data* para uma interface de celular foi utilizado o *MobMuPlat*. Este se trata de um aplicativo para celular que estabelece a comunicação entre a tela do dispositivo e o *Pure Data* por interface amigável ao usuário. Os primeiros resultados experimentais demonstraram que o uso de ferramentas computacionais não só é possível como também acessível a todos.

## 3.3 O papel dos softwares livres na retomada do Cultura Viva

Nessa pesquisa o autor apresenta importância do software livre para produção cultural comunitária no Cultura Viva (LUNA; GAMA, 2024). O Cultura Viva foi uma política criada em 2004 e que virou em 2014, com o objetivo de promover a produção e a difusão cultural no Brasil, por meio de apoio a coletivos. Apesar de promover ações

para integrar a tecnologia com a cultura, diversas limitações foram encontradas, sendo a principal delas a dependência em software proprietário. O grande problema se dava pelo custo da aquisição e licenças de uso e também pela falta de compatibilidade em sistemas que possuem um desempenho limitado.

Os softwares livres surgem como alternativa para mitigar os problemas supracitados. Sistemas de código aberto não cobram pela sua aquisição ou por licença de uso e ainda possui um suporte amplo devido a uma comunidade de desenvolvedores que colaboram em conjunto a fim de melhorar a qualidade e eficiência. O software livre ainda surge como uma solução para a obsolescência programada, que é a ideia de criar softwares ou hardwares com um prazo de validade estabelecido.

Na 4º Conferência Nacional de Cultura (CNC), foi desenvolvida uma página WEB para armazenar textos de material audiovisual. Essa página foi conectada a um perfil que visava a difusão do material no Fediverso. O autor ainda cita o protocolo *ActivityPub* como uma ferramenta para unificar sistemas utilizados no Cultura Viva.

# 4 Projeto

Este capítulo busca detalhar o processo para configurar um servidor e rodar uma instância do Funkwhale. O Funkwhale facilita o processo de configuração do servidor, fornecendo uma documentação simples e objetiva dos passos necessários para rodar a sua própria instância [1]. Apesar de possuir uma documentação objetiva e simples de entender, possui algumas configurações mais avançadas que pode apresentar dificuldades para usuários sem conhecimento de protocolos de rede.

## 4.1 Instalando o servidor Funkwhale

Atualmente o Funkwhale possui suporte para sistemas baseados em Debian [2], sistemas Arch Linux, Docker, entre outros. Docker é uma ferramenta de desenvolvimento que permite você construir e rodar sistemas em containers. Contêiner é uma unidade de software que empacota o código da aplicação e todas suas dependências incluindo tudo o que é necessário para a execução do sistema [3]. O Funkwhale possui um script de instalação para sistemas baseados em Debian que automatiza grande parte das configurações, além disso ele possui a opção de baixar a versão mais recente dos sistema e realizar toda a configuração por conta própria em sistemas Debian, containers do Docker entre outros.

Atualmente o Funkwhale possui três maneiras de acessar a aplicação, pelo aplicativo web, pelo aplicativo do Funkwhale para celular e por aplicativos compatíveis com o Subsonic. A principal maneira de acessar o Funkwhale é por meio da aplicação web escrita em Vue.js e TypeScript, que se comunica diretamente com a API. O aplicativo oficial do Funkwhale está disponível somente para dispositivos android, porém a plataforma também é compatível com aplicativos Subsonic, como o SubStreamer, disponível para android e ios. Independente da forma como o sistema é acessado, as requisições são tratadas pela ferramenta de proxy reverso e mapeadas para a api. A api do Funkwhale é formada pelo Django, que gerencia e trata das requisições, o PostgreSQL, que é o sistema gerenciador de banco de dados, Redis, que serve para fazer caching e acelerar o acesso aos dados da aplicação, Celery Worker, que realiza tarefas de forma assíncrona e o Celery Beat Task, que realiza tarefas agendadas da aplicação, sendo os dois últimos utilizados para tarefas relacionadas a federação. No caso da configuração utilizando o Docker, cada ferramenta que compõe a api é um contêiner no servidor.

[1] Disponível em: https://docs.funkwhale.audio/administrator/index.html
[2] Um sistema operacional de código aberto que é utilizado como base para diversos sistemas Linux (Ubuntu, PureOS, etc.) https://www.debian.org/intro/why_debian
[3] Via https://www.docker.com/resources/what-container/

Para configurar uma instância do Funkwhale é necessária alguma ferramenta de proxy reverso. Proxy reverso é uma camada entre o servidor e o usuário, que fica responsável por receber as requisições do cliente e repassar para o servidor correspondente. A documentação do Funkwhale apresenta o nginx [4] como uma solução de proxy reverso padrão, porém essa configuração pode ser feita por outros serviços, como por exemplo o Apache Server. O nginx ou *Engine X* é uma ferramenta para configurar servidor web HTTP, proxy reverso, cache de conteúdo, balanceador de carga, servidor proxy TCP/UDP e servidor proxy de e-mail. Além de alguma ferramenta de proxy reverso, é necessário ter o Python instalado na máquina e realizar as configurações do Django, além disso também é necessário configurar um banco de dados PostgreSQL para armazenar as informações da aplicação.

O presente projeto visa configurar um servidor HTTPs utilizando o AWS *Lightsail* como servidor dedicado para rodar uma instância do Funkwhale na versão 1.4.0. Para isso será utilizado o Docker para configurar os contêineres que serão executados e o nginx como ferramenta de proxy reverso e o *DuckDns* para configurar um domínio gratuito. O AWS *Lightsail* é uma ferramenta para construir servidores privados virtuais (VPS) para executar aplicações WEB. Esta ferramenta apresenta ambientes já configurados com base nas tecnologias que serão utilizadas, oferecendo instâncias pré-configuradas para Node.js, Django, Nginx, LAMPP, entre outros. Instâncias do *Lightsail* são configuradas através do *Bitnami* [5], que é a ferramenta responsável por pré-configurar os serviços e tecnologias necessários para servidores WEB, essa ferramenta será bastante útil para configurar o *proxy* reverso e o certificado de segurança do sistema, como abordaremos mais pra frente. Para o presente trabalho será utilizada uma instância *Lightsail* pré-configurada para utilizar o nginx, isso permite que as etapas de instalação do nginx não sejam necessárias. Para configurar o domínio (URL) da aplicação é utilizada a plataforma *DuckDns*, esta apresenta uma solução gratuita para registro e configuração de domínios públicos. Para utilizar essa ferramenta gratuita de registro de DNS é necessário criar uma conta no site da plataforma [6] e registrar um domínio que ainda não tenha sido registrado, após registrar o mesmo basta inserir o Ipv4, ou Ipv6 público do servidor da aplicação que o *DuckDns* fica responsável pela parte de registros de DNS. Os domínios do *DuckDns* são subdomínios da plataforma, ou seja, domínios do tipo *nomedodomino.duckdns.org* e no caso do presente trabalho esse será *funkwhalegustavo.duckdns.org*.

Para começar o processo de instalação e configuração do Funkwhale é indicado a criação de um usuário específico para realizar estas ações. A documentação apresenta o comando para criar o usuário chamado Funkwhale e definir a *home* (o diretório padrão do

[4] É um servidor web HTTP que disponibiliza ferramentas de proxy reverso, caching, proxy de email, etc. https://nginx.org/
[5] https://bitnami.com/
[6] https://www.duckdns.org/

usuário) para a pasta */srv/funkwhale/* que é onde ficam os arquivos da aplicação. Após configurar o usuário é possível utiliza-lo para fazer as configurações do servidor. Com o usuário configurado, o próximo passo é clonar o template do docker para a pasta mencionada anteriormente. O template é responsável por informar os detalhes dos containers da aplicação e como os mesmos se comunicam. Após baixar o template do docker é necessário configurar o arquivo de variáveis de ambiente para o Funkwhale, esse arquivo, chamado de *.env* possui as informações da instância, como o endereço do banco de dados e a url da aplicação. Nesse arquivo é necessário definir a versão do Funkwhale (nesse caso é a 1.4.0), uma chave secreta para o Django, que pode ser criada pela biblioteca *openssl* disponível para windows e linux. Também é necessário configurar o host da aplicação, que é o url da mesma, que nesse caso será *funkwhalegustavo.duckdns.org* para rodar localmente. No caso das configurações de banco de dados e *caching* não é necessário realizar alguma mudança nas configurações.

Após configurar as variáveis de ambiente é necessário baixar todos serviços necessários pelos contêineres, isso pode ser feito pelo comando *docker compose pull*. Após baixar todos os serviços essenciais é necessário iniciar o banco de dados para rodar o comando de migração de banco de dados, que cria todas as tabelas e colunas que serão necessárias para o sistema. Após rodar as migrações do banco de dados já é possível criar um usuário com privilégios. Esse usuário possuirá privilégios dentro da aplicação e terá acesso às ferramentas de administração de usuários e conteúdo, bem como gerenciar as permissões e informações das instâncias. Após realizar essa configuração já é possível executar os contêineres do docker para iniciar a aplicação.

Para gerenciar a comunicação entre o cliente e o servidor da aplicação é necessário configurar uma ferramenta de proxy reverso. Como mencionado, será utilizado o nginx para realizar essa configuração, portanto o primeiro passo nessa etapa é instalar o nginx na máquina para começar a configuração. Como é utilizada uma instância do *Lightsail* com o nginx pré-configurado, a etapa de instalação do mesmo pode ser ignorada partindo direto para a etapa de configuração. Para configurar o *proxy* reverso é necessário baixar o template de configuração do nginx para o Funkwhale. Esse template possui os campos necessários para que o nginx consiga fazer o proxy reverso da aplicação. Após baixar esse arquivo basta atualizar o mesmo com as informações da sua aplicação, que nesse caso são o host e a versão do Funkwhale, e criar um link simbólico desse arquivo com a pasta *sites-enabled* dentro da pasta do nginx. Essa pasta possui os links simbólicos para os arquivos da pasta *sites-available*, essa segunda pasta contém as informações de configuração do nginx da aplicação.

O último passo necessário para colocar uma instância do Funkwhale no ar é configurar o TLS. O *Transport Layer Security* (TLS) é um protocolo de segurança que criptografa a comunicação entre o cliente e o servidor da aplicação. O TLS é necessário para

rodar a instância no protocolo HTTPS que possui uma camada adicional de segurança. Para configurar o TLS é necessário gerar um certificado para a aplicação, na documentação do Funkwhale é indicado o uso do *Certbot* [7] para esta tarefa. O *Cerbot* gera um certificado *LetsEncrypt* que é uma autoridade de certificado (CA) gratuita, automática e de código aberto, que é fornecida pelo *Internet Security Research Group* (ISRG), uma organização sem fins lucrativos dedicada a melhorar a segurança na internet [8]. Para gerar um certificado *Lestencrypt* podemos utilizar o *Certbot*, porém neste caso utilizaremos a ferramenta de geração de certificados disponibilizada pelo *Bitnami*. Para isso é necessário instalar o *Lego Client*, esta se trata de uma ferramenta para geração e renovação de certificados *LetsEncrypt*. Para instalar o *Lego* e gerar o certificado basta seguir os passos presentes na documentação do *Bitnami* [9]. Após a instalação, o arquivo de geração de certificados fica no diretório */opt/bitnami/letsencrypt/lego* e basta executar o mesmo passando o email que receberá avisos sobre renovação do certificado, os domínios que serão utilizados (geralmente são passados dois domínios, um com o www e um sem, porém nesse caso será utilizado somente o sem) e o caminho no qual os certificados serão salvos. Após executar o *Lego Client* são gerados dois arquivos, um com a extensão .crt que é o certificado e outro .key que é a chave privada do certificado, para os dois são feitos links simbólicos (que é um arquivo que aponta para outro arquivo) com os arquivos *server.crt* e *server.key* localizados no diretório */opt/bitnami/nginx/conf/bitnami/certs/* (que é onde os certificados são salvos no *Bitnami*). Por último basta adicionar o certificado e a chave privada no arquivo de configuração do nginx para o *Funkwhale* e reiniciar a ferramenta de *proxy* reverso, para ser possível acessar o domínio a partir do protocolo HTTPS.

Para federar uma instância do Funkwhale, é necessário ativar a opção de federação nas configurações da instância (por padrão essa opção já vem ativada), garantir que ela tenha um certificado SSL válido e que esteja com a visibilidade definida como pública. A federação na plataforma funciona por meio das bibliotecas: ao publicar conteúdo dentro de uma biblioteca, o usuário tem a opção de seguir bibliotecas remotas. Para isso, basta colar o link de uma biblioteca de outra instância do Funkwhale. Caso a biblioteca esteja configurada como pública para todas as instâncias, o seu conteúdo poderá ser acessado e reproduzido em qualquer pod federado, ampliando o alcance e a disponibilidade da mídia compartilhada na rede. Como mencionado anteriormente, o Celery Worker e Beat Task são responsáveis pelas tarefas de federação, fazendo leituras nas bibliotecas remotas para atualizar o conteúdo da biblioteca com base na original.

[7] O *Certbot* é uma solução gratuita e de código aberto para a geração de certificados para habilitar o protocolo HTTPS

[8] https://www.abetterinternet.org/about/

[9] https://docs.bitnami.com/general/how-to/generate-install-lets-encrypt-ssl/

## 4.2 Configurando a aplicação

Uma instância do Funkwhale pode ser configurada pelo arquivo de variáveis de ambiente ou pela tela de configuração do pod dentro da plataforma. O arquivo de variáveis de ambiente contém configurações relacionadas à arquitetura da aplicação, permitindo alterar a ferramenta de proxy reverso, configurar o banco de dados e o Redis, ajustar a API, definir parâmetros de federação, entre outras opções avançadas. Por mais que seja possível alterar o host da aplicação a qualquer momento pelo arquivo de configuração, essa alteração faz com que o sistema pare de funcionar, o que ocorreu neste projeto durante a primeira tentativa de publicar uma instância. A tela de configuração do Funkwhale, acessível pela interface web, permite personalizar aspectos operacionais e administrativos da instância sem a necessidade de edição manual do ‘.env‘. Por meio dela, é possível definir informações da instância, como nome, descrição e email de contato, configurar permissões de usuários, habilitar ou desabilitar o cadastro aberto, ajustar cotas de upload (que por padrão é de 1GB), habilitar ou desabilitar a federação, controlar a visibilidade do conteúdo e definir regras de moderação. Além disso, há opções para personalizar a interface do usuário, ativar ou desativar a API Subsonic, habilitar ou desabilitar canais de músicas e gerenciar limites de canais e número de músicas em uma playlist por usuário.

A configuração da instância também inclui opções avançadas, como a habilitação de transcodificação de arquivos de áudio, que permite a conversão de formatos não suportados por determinados dispositivos, e a definição de um tempo de cache para arquivos transcodificados. Outra funcionalidade importante é a possibilidade de exigir que os arquivos de áudio sejam marcados com um identificador do MusicBrainz, o que facilita a organização e a catalogação do conteúdo. Além disso, a interface de configuração permite ajustar o tamanho máximo de playlists, o número máximo de canais por usuário e a duração do cache de músicas federadas.

Por fim, a tela de configuração oferece ferramentas para moderação, como a habilitação de listas de permissões (*allow-listing*) para controlar a federação com outras instâncias, e a definição de categorias de denúncias que podem ser enviadas por usuários anônimos. Também é possível personalizar mensagens de suporte, adicionar regras de conduta e termos de serviço, e até mesmo inserir código CSS personalizado para alterar a aparência da interface. Essas opções tornam o Funkwhale uma plataforma altamente flexível, capaz de se adaptar às necessidades específicas de cada comunidade.

## 4.3 Configurando os clientes

Como mencionado anteriormente, é possível acessar o Funkwhale por meio de três clientes diferentes, a aplicação web, o aplicativo oficial da plataforma e aplicativos Subsonic. Vale ressaltar que é necessário publicar uma instância na web para ser capaz

de utilizar os aplicativos e as configurações da instância são feitas exclusivamente na aplicação web.

## Aplicativos Subsonic

O aplicativo oficial do Funkwhale está disponível somente para sistemas android, porém ainda é possível acessar uma instância do Funkwhale em outros sistemas por meio de aplicativos com suporte a api do Subsonic. Subsonic se trata se uma ferramenta que permite reproduzir músicas de forma remota. Para IOS existe o aplicativo SubStreamer que permite logar remotamente em servidores Subsonic e ouvir as músicas por meio do aplicativo. O Funkwhale possui suporte ao Subsonic e o mesmo pode ser configurado na tela de configurações da instância, podendo ser ativado ou desativado (o suporte ao Subsonic já vem ativado por padrão).

Para acessar o conteúdo de uma instância do Funkwhale por meio de um aplicativo compatível com Subsonic, é necessário configurar uma senha específica para a API dentro da plataforma. Essa opção está disponível nas configurações do usuário, onde é possível gerar uma senha exclusiva para uso com aplicativos Subsonic. A senha gerada é aleatória, composta por várias palavras em inglês, e deve ser utilizada para autenticação. Para efetuar o login em um aplicativo Subsonic, o usuário deve fornecer seu nome de usuário do Funkwhale e a senha gerada para a API. Após a autenticação, o conteúdo da instância será carregado e poderá ser acessado diretamente no aplicativo.

## Cliente no browser

Atualmente, o Funkwhale conta com somente um cliente web oficial. Este é o cliente padrão do Funkwhale e é a partir deste que são realizadas as configurações da instância. Por mais que exista somente um cliente no browser, é possível customizar a interface da instância por meio de estilizações CSS. Dentro da tela de configuração do admin existe a opção de adicionar uma estilização CSS personalizada, permitindo alterar as cores e fontes da instância, permitindo diferenciar a interface da instância para as demais. Além de estilizações personalizadas, também é possível alterar os textos da plataforma, alterando o nome da instância e a descrição, além de permitir a alteração dos textos de login e cadastro.

O Funkwhale possui um conjunto de permissões que definem a maneira como cada usuário interage com o sistema. As permissões dos usuários podem ser designadas no painel de administração do Django ou dentro da tela de moderação da plataforma. É possível adicionar o status de admin para os usuários, o que permite o acesso a todas as funcionalidades do sistema, permitindo excluir músicas e bibliotecas, acessar o painel de moderação e acessar o painel de configuração da instância. Também é possível designar

permissões específicas para cada usuário. As permissões disponíveis são em relação a moderação, configuração da instância e gerenciamento de bibliotecas. As permissões de moderação permitem que um usuário revise e gerencie conteúdos enviados à instância, podendo gerenciar denúncias e aplicar ações aos usuários, como excluir faixas ou restringir usuários que violem as diretrizes da plataforma. Já as permissões de configuração da instância garantem acesso a tela de configuração da instância, permitindo ajustes a nível global, como ativar ou desativar a federação, modificar políticas de acesso e personalizar a aparência e o funcionamento do Funkwhale. Por fim, as permissões de gerenciamento de bibliotecas permitem que um usuário crie, edite e exclua bibliotecas, controle a visibilidade do conteúdo e gerencie colaborações, determinando quais usuários podem adicionar ou remover músicas de uma biblioteca específica.

A organização das músicas na aplicação web é feita a partir de canais, bibliotecas, playlists, rádios e favoritos. Canais são utilizados por usuários que desejam publicar conteúdo autoral na plataforma. Existem dois tipos de canais na plataforma, o canal de artista, que é utilizado para publicar músicas de um determinado artista e canal de podcast, que como o nome indica é utilizado para publicar podcasts de algum autor específico. O Funkwhale disponibiliza para os usuários a opção de criar canais, edita-los e exclui-los, além de apresentar a opção de se inscrever no canal por meio de feeds RSS e inscrever em feed de podcasts para serem transmitidos para dentro da plataforma. A biblioteca é destinada para que os usuários publiquem suas músicas favoritas na plataforma, disponibilizado diferentes opções de visibilidade. As opções disponíveis são: visível para todos em todas as instâncias, permitindo que qualquer usuário de qualquer pod federado acesse a biblioteca; visível para todos nesta instância, restringindo o acesso apenas aos usuários da mesma instância; e visível somente para mim, tornando a biblioteca privada e acessível apenas ao próprio usuário que a criou. As playlists são listas de músicas criadas pelos usuários. Ao criar uma playlist é possível definir a visibilidade para todos os usuários de todas as instâncias, todos os usuários da instância atual e somente para o próprio usuário. Playlists possuem um limite máximo de músicas, o que pode ser definido pelo administrador e por padrão é de 250 músicas. As rádios são playlists dinâmicas construídas pela aplicação com base em algum critério específico. Existem cinco tipos de rádio no Funkwhale, uma que gera uma playlist com base na músicas das suas bibliotecas, uma que consiste da sua lista de favoritos, uma que gera uma lista completamente aleatória, uma baseada nas últimas músicas adicionadas na plataforma e a última gerando uma lista de reprodução com base nas músicas com menos reproduções.

# 5 Resultados

Após realizar as configurações necessárias apresentadas na seção anterior, foi possível executar uma instância do Funkwhale de forma local na máquina. Ao acessar a url *https://funkwhalegustavo.duckdns.org* é apresentada a tela inicial da aplicação, que apresenta algumas informações básicas sobre a instância e um formulário de login, além de mostrar o menu lateral que apresenta opções para navegar pelas músicas, artistas, albúns e playlists presentes na instância. A figura abaixo mostra a tela inicial do Funkwhale.

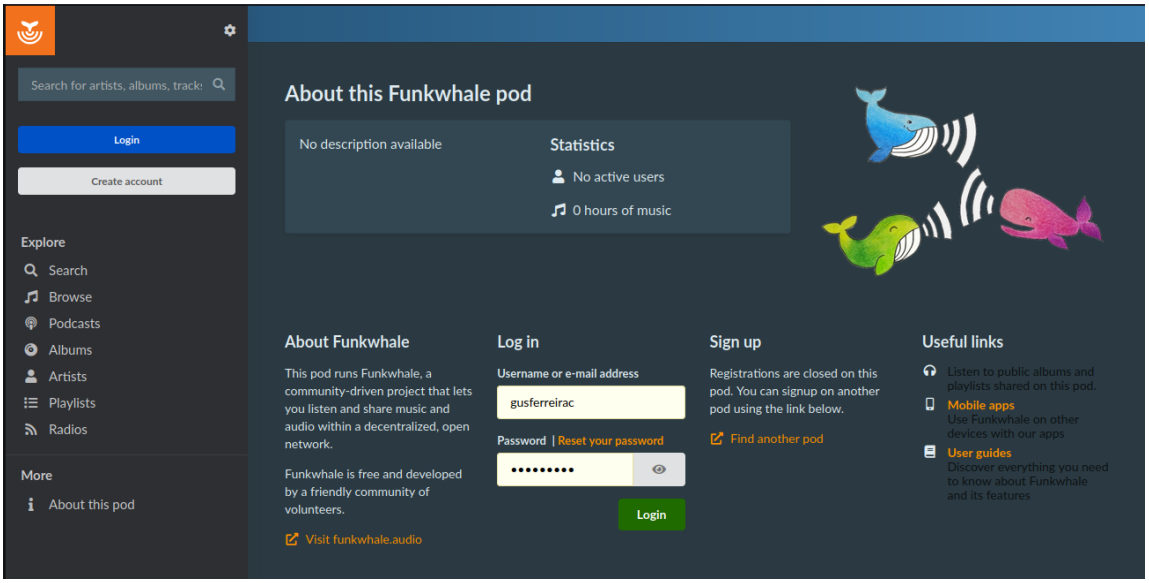


Figura 7 – Tela Inicial do Funkwhale

É possível acessar o sistema a partir do usuário criado anteriormente na configuração. Como dito anteriormente, este usuário possui permissões de administrador. Por possuir permissões de administrador é apresentado um ícone de uma chave de boca no campo superior esquerdo que apresenta uma opção de configurar a instância, onde apresente opções para definir um nome e uma descrição para a instância, bem como opções para gerenciar a privacidade da instância, gerenciar as bibliotecas e os conteúdos das mesmas. Além disso também apresenta uma opção para gerenciar todos os usuários do pod e uma aba de moderação que é onde aparece os reports feitos pelos usuários dentro da instância. A figura 8 mostra a tela de configuração de instância, que apresenta campos para definir o nome, a localização, uma descrição curta e longa, um email de contato, uma lista de regras, os termos de serviço, uma imagem de banner e uma mensagem que será mostrada periodicamente para os usuários (criada com o intuito de solicitar doações para manter o servidor da aplicação).

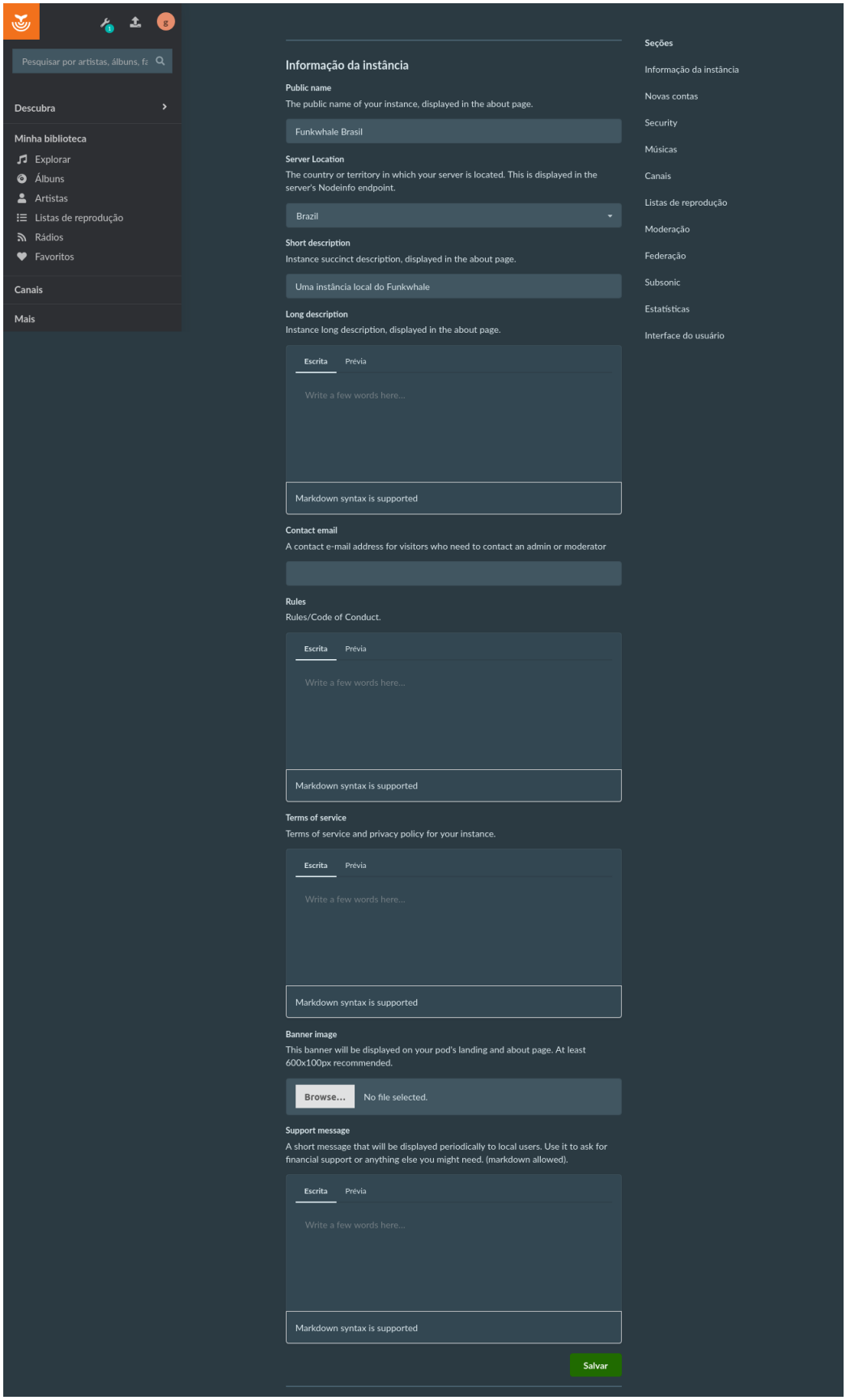


Figura 8 – Tela de configuração de instância

Além de apresentar opções gerais da instância, a tela de configuração da mesma ainda possui uma seção para configurar o cadastro de contas novas, permitindo privar ou liberar o acesso à aplicação, uma seção para definir a segurança, permitindo configurar as permissões dos usuários e definir o limite de *upload* dos mesmos, uma sessão de músicas e canais para configurar ambos e outras seções para configurar a moderação, a federação, o plugin *Subsonic*, estilização personalizada e interface do usuário.

Quando um usuário faz algum *report* (denúncia), de algum conteúdo ou usuário que viola as regras da plataforma é enviada uma notificação para os administradores da plataforma. Essa notificação aparece no ícone de gerenciamento da instância e a denúncia pode ser acessada pelo painel de moderação, onde é informada a conta que fez a denúncia, o objeto da denuncia (outro usuário, música ou biblioteca de músicas) e o tipo de *report* que está sendo feito. A figura 9 apresenta a tela de denúncias contendo uma denúncia de conteúdo ilegal em uma biblioteca.

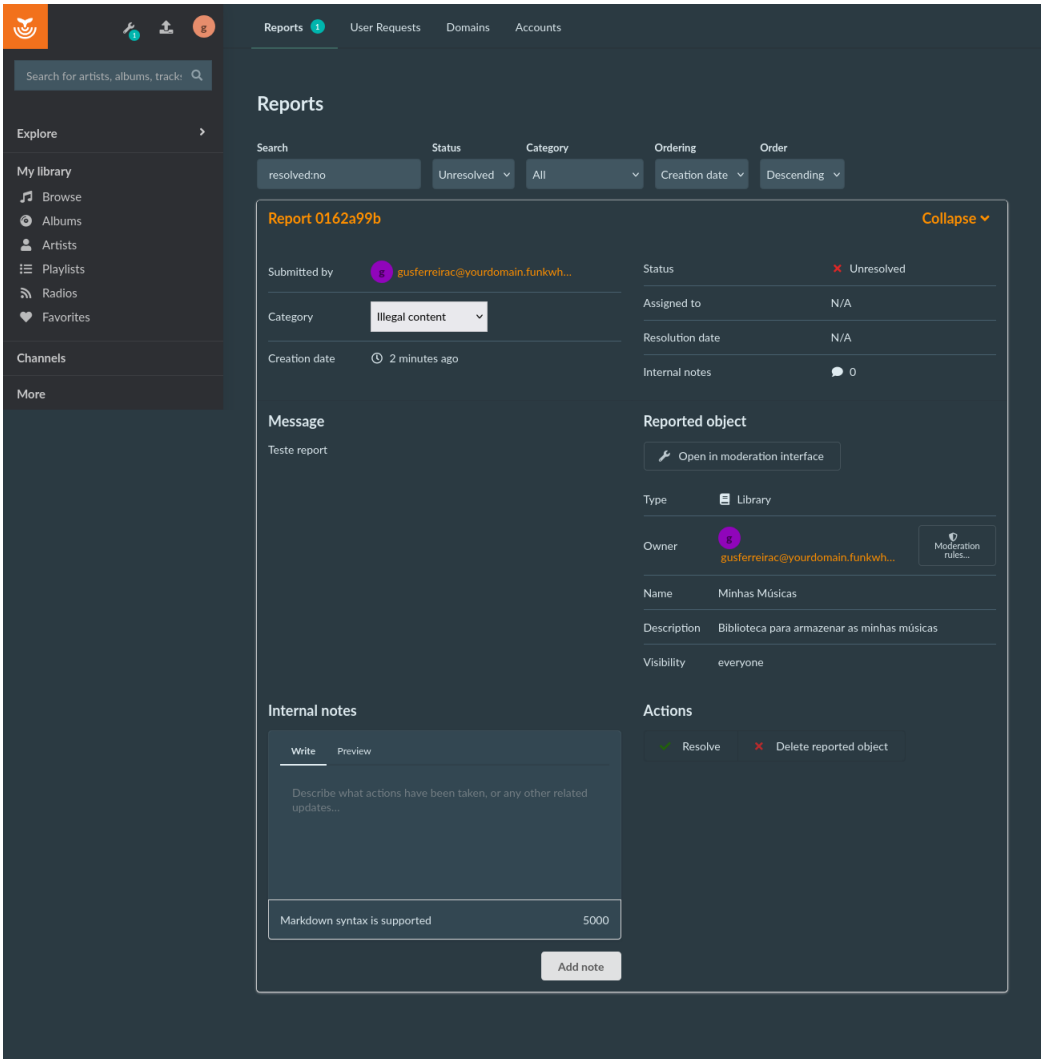


Figura 9 – Tela de denúncias

A plataforma disponibiliza três opções para os usuários publicarem suas músicas, a primeira é de disponibilizar a música em um canal público (opção destinada para músicos e *podcasters*), adicionar músicas em sua biblioteca pessoal e a opção de seguir bibliotecas remotas de outros servidores. Um detalhe importante é que para conseguir publicar suas músicas em uma instância do *Funkwhale* é necessário que o arquivo que está sendo enviado possua alguns metadados para que música seja processada corretamente na plataforma. A figura 10 apresenta a tela de biblioteca de usuário, que mostra as músicas presentes na mesma além de mostrar opções de *upload* e personalização da biblioteca.

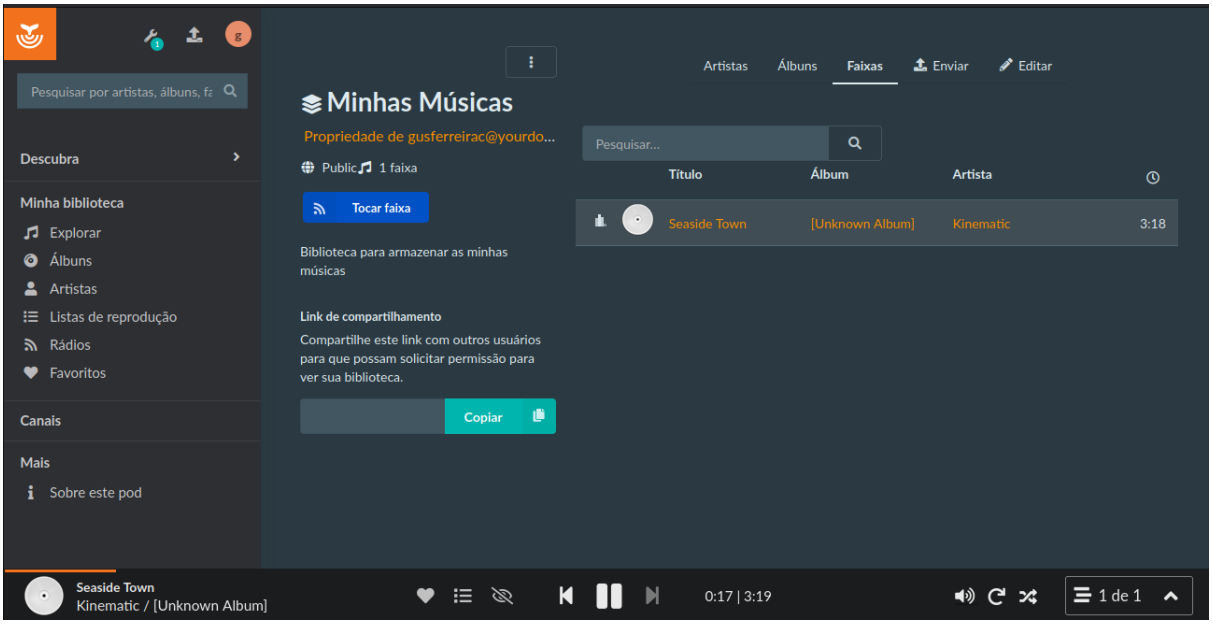


Figura 10 – Biblioteca de músicas

Ao escolher a opção de seguir uma biblioteca remota é apresentado ao usuário um campo para inserir a URI da mesma. Ao inserir o identificador da biblioteca a mesma é exibida e é mostrado uma opção para enviar uma solicitação para segui-la desde que a sua visibilidade esteja pública para todas as instâncias. Ao enviar essa solicitação, o proprietário da biblioteca recebe uma notificação da solicitação e possui a opção de aprovar ou não, caso aprovada o usuário que está seguindo terá acesso ao conteúdo da biblioteca na sua instância e poderá escaneá-la periodicamente para atualizar o seu conteúdo. Para testar a federação foi feito o cadastro de uma conta na plataforma *The Funky Boi*, uma instância do Funkwhale que permite o cadastro e o acesso sem necessidade de aprovação, após a criação da conta a biblioteca da instância criada nesse trabalho foi adicionada e a partir disso foi possível acessar o conteúdo da mesma e escutar as músicas em outro domínio. A figura 11 apresenta a tela para adicionar bibliotecas remotas e a biblioteca escaneada da instância *funkwhalegustavo.duckdns.org*, como é possível observar um *scan* está sendo feito no momento e se houver uma diferença entre a biblioteca remota e a

original o conteúdo da versão remota será atualizado.

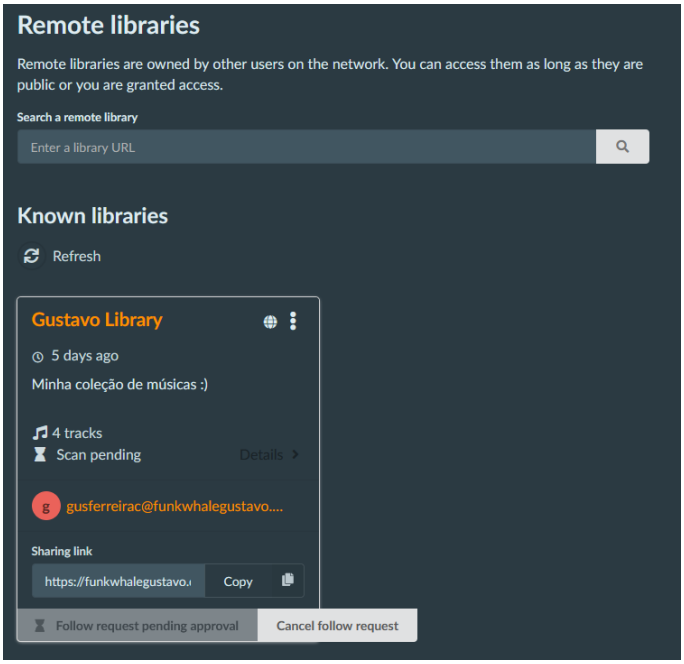


Figura 11 – Tela para adicionar bibliotecas remotas

A interface da aplicação permite a alteração da estilização do sistema, permitindo que os administradores insiram classes CSS específicas para alterar a aparência do sistema. O tema *Dracula* [1] presente em diversas aplicações, como Visual Studio e Notepad++, possui uma versão para o Funkwhale. A versão do tema para o Funkwhale é disponibilizada a partir de um código css que pode ser aplicado na tela de configuração de instância. Aplicamos o tema à instância, porém com algumas alterações nas cores alterando a paleta do tema para uma baseada em tons azuis. A imagem 12 apresenta a opção para inserir css personalizado com o tema aplicado.

[1] https://draculatheme.com/


Figura 12 – Estilização personalizada

Na tela de configurações do usuário é possível estabelecer conexões entre a plataforma e plugin e/ou aplicações externas. Dentro da tela de configuração é possível gerar uma senha para ser utilizada por aplicativos Subsonic. A senha gerada é composta de cinco palavras em inglês em uma ordem aleatória. Essa senha é necessária para acessar a instância a partir desses aplicativos, já que a senha utilizada na aplicação web não serve nesse caso. A imagem 13 apresenta a opção de gerar as credenciais para utilizar nas aplicações e a imagem 14 apresenta a tela inicial da aplicação SubStreamer após realizar o login na instância.

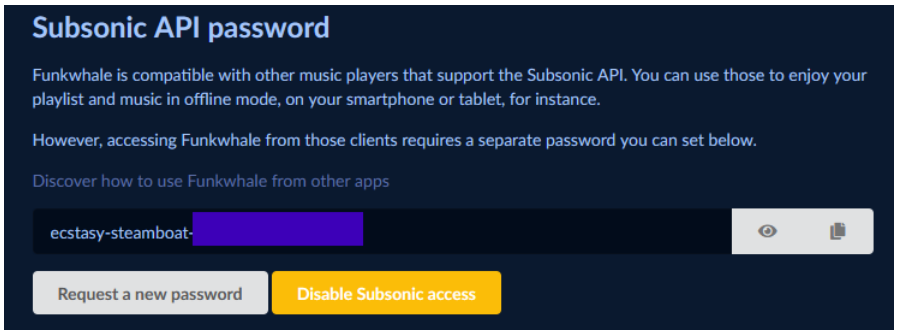


Figura 13 – Geração de senhas para o Subsonic

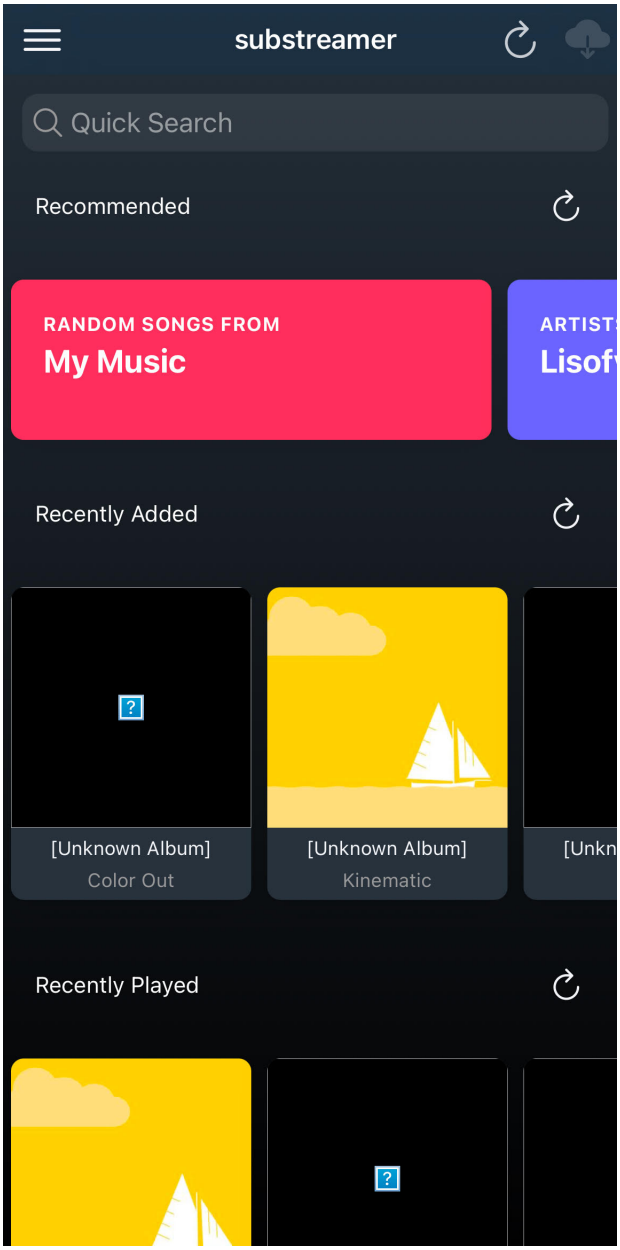


Figura 14 – Tela inicial do SubStreamer

Como mencionado anteriormente o Funkwhale possui um aplicativo oficial disponível para android. O aplicativo é disponibilizado de maneira gratuita dentro da *Play Store* (loja de aplicativos do android). O aplicativo atualmente possui mais de cem mil downloads e é atualizado frequentemente com melhorias, inclusão de novas funcionalidades e correções de bugs. A imagem 15 apresenta a tela de download do aplicativo na loja de aplicativos do android.


Figura 15 – Aplicativo oficial do Funkwhale

# 6 Conclusão e Trabalhos Futuros

O presente trabalho apresentou a arquitetura principal do Fediverso e principalmente da plataforma de código aberto Funkwhale bem como detalhou o processo para clonar o código-fonte da aplicação e executar o mesmo em um servidor local utilizando o Docker e o nginx. O Funkwhale possui uma interface simples e intuitiva que se assemelha à de outras plataformas de *streaming* de áudio, o que ajuda usuários já acostumados a essas outras plataformas a se ambientar na nova interface. Além disso, a documentação da plataforma possui uma documentação objetiva e com um processo de instalação relativamente simples, devido a grande parte da configuração do sistema ser configurada automaticamente. Esta plataforma demonstrou ser uma excelente alternativa às plataformas proprietárias visto a simplicidade das suas telas e a velocidade do sistema, além de transformar o processo de *upload* e divulgação de músicas em algo simples, sem a necessidade de intermediadores, o que faz com que esta plataforma seja uma ótima alternativa para artistas independentes.

A relevância deste estudo vai além da análise técnica da plataforma, pois reforça a necessidade de redes sociais descentralizadas para a distribuição de conteúdo cultural. No contexto da música independente e do movimento *underground*, serviços proprietários, como Spotify e YouTube, podem apresentar barreiras significativas para os artistas, seja pelo alto custo de distribuição, pela censura de conteúdo ou pela aplicação rigorosa de leis de direitos autorais, impedindo a divulgação de trabalhos que utilizam samples ou abordam temas não aceitos por essas plataformas. Nesse cenário, uma solução descentralizada como o Funkwhale permite que artistas publiquem e compartilhem suas obras sem intermediários, conectando-se diretamente com seu público.

Se essa plataforma se popularizasse, poderíamos imaginar um cenário em que a distribuição de música se tornaria mais acessível, democratizando o acesso à produção cultural. Artistas independentes poderiam construir comunidades em torno de sua arte, sem depender de algoritmos que priorizam grandes gravadoras ou conteúdo patrocinado. Além disso, a resistência à censura e o controle total sobre o próprio conteúdo transformariam o Funkwhale em um ambiente mais livre e colaborativo.

## 6.1 Dificuldades Encontradas

Apesar de possuir uma documentação objetiva e automatizar grande parte das configurações do sistema, o processo de colocar uma instância do Funkwhale no ar não é trivial. Por não possuir muito conhecimento prévio a este trabalho sobre ferramentas de *proxy* reverso e contêineres do Docker tive um pouco de dificuldade para configurar o

Docker e principalmente, configurar o nginx da forma correta. Outra dificuldade encontrada foi na geração do certificado TLS, pois não consegui utilizar a ferramenta *Certbot* presente na documentação do Funkwhale e tive que buscar uma alternativa gerando um certificado com a ferramenta presente no *Bitnami*.

Outro detalhe importante é que como mencionado anteriormente no trabalho, a músicas devem conter alguns metadados para que possa ser processada corretamente dentro da plataforma. Quando o arquivo de aúdio não possui as tags corretamente, que nesse caso são título da música e artista, o *upload* apresenta um erro que não indica de maneira clara qual foi o erro encontrado, ficando a cargo do usuário descobrir o erro e tentar resolvê-lo. Isso não se apresenta como uma falha grave mas sim um pequeno problema de usabilidade que deve ser corrigido nas próximas versões.

A federação também apresentou alguns problemas, onde em certas instâncias não foi possível adicionar uma biblioteca remota mesmo com as configurações de federação ativadas. Além disso em certos casos as músicas de bibliotecas remotas desapareciam com o tempo e mesmo escaneando novamente as mesmas as músicas não apareciam novamente. A hipótese levantada é que alguns configurações ainda não estão funcionando completamente visto que a plataforma está em constante desenvolvimento, porém as funcionalidades principais, como *upload* e reprodução de músicas funcionam corretamente não apresentando erros.

## 6.2 Trabalhos Futuros

O presente projeto foi capaz de executar uma instância do Funkwhale em uma rede local. Para um trabalho futuro seria interessante configurar uma instância em uma rede pública, permitindo o acesso de usuários do mundo inteiro e federando a plataforma para comunicar com outras instâncias existentes.

# Referências

ANDERSON, A. et al. Algorithmic effects on the diversity of consumption on spotify. In: *Proceedings of the web conference 2020*. [S.l.: s.n.], 2020. p. 2155–2165. Citado na página 12.

BADAWY, A.; FERRARA, E.; LERMAN, K. Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign. In: *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.: s.n.], 2018. p. 258–265. Citado na página 10 (2 ocorrências).

CAUFFMAN, C.; GOANTA, C. A new order: The digital services act and consumer protection. *European Journal of Risk Regulation*, Cambridge University Press, v. 12, n. 4, p. 758–774, 2021. Citado na página 11.

CAVA, L. L.; GRECO, S.; TAGARELLI, A. Understanding the growth of the fediverse through the lens of mastodon. *Applied Network Science*, Springer Science and Business Media LLC, v. 6, n. 1, set. 2021. ISSN 2364-8228. Disponível em: <http://dx.doi.org/10.1007/s41109-021-00392-5>. Citado na página 10.

CERRUTO, F. et al. Social network data analysis to highlight privacy threats in sharing data. *Journal of Big Data*, Springer, v. 9, n. 1, p. 19, 2022. Citado na página 10.

LUNA, C. E. F.; GAMA, J. R. O papel dos softwares livres na retomada do cultura viva. ENECULT, 2024. Citado na página 27.

MANSOUX, A.; ABBING, R. R. Seven theses on the fediverse and the becoming of floss. Institute for Network Cultures and Transmediale, 2020. Citado nas páginas 11 e 14 (2 ocorrências).

OTT, B. L. The age of twitter: Donald j. trump and the politics of debasement. *Critical studies in media communication*, Taylor & Francis, v. 34, n. 1, p. 59–68, 2017. Citado na página 10 (2 ocorrências).

PRODROMOU, E. *OpenMicroBlogging*. 2008. Https://www.w3.org/2008/09/msnws/papers/W3C_FOSN_Position_Paper. Citado na página 15.

RAO, P. R. M.; KRISHNA, S. M.; KUMAR, A. S. Privacy preservation techniques in big data analytics: a survey. *Journal of Big Data*, Springer, v. 5, n. 1, p. 33, 2018. Citado na página 10.

RIBEIRO, J. de F. et al. O uso da tecnologia para práticas musicais em grupos de percussão. In: *Anais do XXXIV Congresso da ANPPOM*. Salvador/BA: ANPPOM, 2024. p. 1–13. ISSN 1983-5973. Citado na página 27.

RIEMANN, R. *Tech Dispatch Federated Social Media Platforms*. [S.l.], 2022. Citado na página 11.

SOUSA, J. C.; SCHIAVONI, F. L. Depurando o underground: artistas independentes capacitando-se em produção musical com software livre. In: *Anais do XXXIII Congresso da ANPPOM*. [S.l.]: ANPPOM, 2023. p. 1–16. ISSN 1983-5973. Citado nas páginas 12 e 26.