INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL
CAMPUS RIO GRANDE
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PIERRE RODRIGUES CALADO

Enkibot: Um Software de Gerenciamento para Streamers da Twitch

Trabalho de Conclusão de Curso
apresentado como requisito parcial
para a obtenção do grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas

Prof. Dr. Igor Avila Pereira
Orientador

Rio Grande, dezembro de 2021

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Calado, Pierre Rodrigues

Enkibot: Um Software de Gerenciamento para Streamers da Twitch / Pierre Rodrigues Calado. – Rio Grande: Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, 2021.

47 f.: il.

Trabalho de Conclusão de Curso (tecnólogo) – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Rio Grande, BR–RS, 2021. Orientador: Igor Avila Pereira.

1. Twitch. 2. Streamer. 3. Bot. 4. Chatbot. I. Pereira, Igor Avila. II. Título.

FOLHA DE APROVAÇÃO

Monografia sob o título *"Enkibot: Um Software de Gerenciamento para Streamers da Twitch"*, defendida por Pierre Rodrigues Calado e aprovada em 10 de dezembro de 2021, em Rio Grande, RS, pela banca examinadora constituída pelos professores:

Prof. Dr. Igor Avila Pereira
Orientador

Prof. MSc. Márcio Josué Ramos Torres
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul

---

Prof. Dr. Eduardo Wenzel Brião
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul

"A persistência é o menor caminho do êxito."
— CHARLES CHAPLIN

AGRADECIMENTOS

Agradeço, primeiramente, a meus amigos por todo incentivo e apoio no desenvolvimento deste trabalho. Além disso, gostaria de agradecer ao meu orientador Igor Avila Pereira pelo auxílio fundamental na elaboração deste projeto, bem como aos demais professores que colaboraram com seus ensinamentos no meu desenvolvimento pessoal e profissional. Por fim, agradecer a minha namorada Sâmia, por me acompanhar nos piores e nos melhores momentos e por todo apoio dado até o último segundo do desenvolvimento deste trabalho.

RESUMO

Com o surgimento de plataformas capazes de permitir que qualquer pessoa possa ter seu próprio canal de tv, mediante a publicação de conteúdo gravado e a criação de lives, surge a necessidade da construção de ferramentas de apoio no que tange o controle de inscritos, o controle de solicitações dos espectadores, etc, principalmente, quando o mais importante é a interação entre os donos destes canais (*streamers*) e seus espectadores no chat durante as transmissões ao vivo. Assim, este trabalho propõe o desenvolvimento de um chatbot, denominado Enkibot, capaz de gerenciar filas de prioridade para os pedidos de revisão de partidas (replays) realizadas pelos espectadores, onde o foco é receber o feedback de um jogador experiente (*streamer*). Além de ser uma ferramenta de apoio para o *streamer*, pois preserva a ordem das solicitações, o Enkibot permite gerenciar todas as filas criadas através de um *dashboard* totalmente gráfico. O chatbot foi desenvolvido em Node.js, React e Json devido à sua capacidade de propiciar o desenvolvimento de aplicações *open-source* multiplataforma e de fácil incorporação por parte da plataforma da Twitch.

Palavras-chave: Twitch. Streamer. bot. chatbot.

LISTA DE SIGLAS E ABREVIATURAS

AHK	*AutoHotkey*
CSS	*Cascading Style Sheets*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*
UUID	*Universally Unique Identifier*

LISTA DE FIGURAS

2.1 Menu de criação de enquetes na Twitch. 19
2.2 Menu dos Pontos de Canal na Twitch. 19

3.1 Twitch Queue Bot: Interface de Configuração. 22
3.2 Twitch Queue Bot: Interface de Estado da Fila. 22
3.3 Play With Viewers: Interface para o controle de filas e chat do canal da Twitch. 23
3.4 Play With Viewers: Modo aleatório e Recurso de *bits*. 24
3.5 Play With Viewers: Telas de Configuração. 24
3.6 Play With Viewers: Recursos Adicionais do Plano Pago (*Premium*). . 25
3.7 Warp World: *Dashboard* da fila de espectadores. 26
3.8 Warp World: Tela de configuração da fila. 26
3.9 Warp World: Página de configuração do visual da *Warp Bar*. 27
3.10 Warp World: Página de configuração da *Warp Bar*. 27

4.1 Diagrama de Casos de Uso. 29
4.2 Dashboard do Enkibot. 31
4.3 Arquitetura do Chatbot. 32
4.4 Tela de exemplo de configuração do arquivo Config.cfg. 34
4.5 Tela de exemplo de uma fila armazenada no arquivo Data.json. 35

6.1 Tela do terminal exibindo o comando de inicialização do Enkibot. . . 38
6.2 Tela da página da Twitch pela visão do streamer. 39
6.3 Chat da Twitch com o comando de exibição de fila com uma fila vazia. 39
6.4 Dashboard exibindo uma fila vazia. 39
6.5 Chat da Twitch com o comando para adicionar o usuário 'Espectador123' à fila. 40
6.6 Streamer utilizando o comando de exibição de fila no chat da Twitch, exibindo o usuário que foi adicionado à fila. 40
6.7 Dashboard exibindo o usuário que foi adicionado à fila. 40
6.8 Outro usuário utilizando o comando para adicionar um replay na fila. 40
6.9 Chat da Twitch exibindo 2 usuários na fila de espera. 41
6.10 Dashboard exibindo os 2 usuários na fila de espera por ordem de chegada. 41
6.11 Chat da Twitch exibindo o usuário inscrito sendo adicionado ao inicio da fila. 41
6.12 Chat da Twitch mostrando a fila com os 3 usuários. 41
6.13 *Dashboard* mostrando a fila com os 3 usuários. 42

6.14 Usuário usando comando para descobrir sua posição na fila no chat da Twitch. 42
6.15 Streamer utilizando o comando para chamar o próximo da fila. 42
6.16 Dashboard mostrando os 2 usuários restantes. 42
6.17 Streamer utilizando o comando de fechar a fila. 43
6.18 Outro usuário utilizando o comando de adicionar replay, mas sendo barrado pois a fila está fechada. 43
6.19 Streamer utilizando o comando de limpar a fila e em seguida o de mostrar a fila, com ela estando vazia. 43
6.20 *Dashboard* com a fila vazia novamente. 43

LISTA DE TABELAS

3.1 Tabela Comparativa dos Sistemas Existentes. 28

SUMÁRIO

1 INTRODUÇÃO 13
1.1 Objetivos 14
1.2 Organização do Texto 14

2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 Bot 15
2.1.1 Chatbot 15
2.2 Streaming 16
2.2.1 Plataformas de *Streaming* de *Games* 16
2.2.2 Streamer e Audiência 16
2.3 Monetização 17
2.3.1 Formas de Monetização 17
2.4 Ferramentas de Apoio para Streamers 17
2.5 Tecnologias de Desenvolvimento 20
2.5.1 Node.js 20
2.5.2 tmi.js 20
2.5.3 React 20
2.5.4 AutoHotkey 20

3 SISTEMAS EXISTENTES 21
3.1 Twitch Queue Bot 21
3.2 Play with Viewers 23
3.3 Warp World Multi-Queue 25
3.4 Comparativo entre os Sistemas Existentes 28

4 PROJETO 29
4.1 Diagrama de Casos de Uso 29
4.1.1 Atores 29
4.1.2 Pacotes 30
4.2 Arquitetura 31
4.2.1 Chatbot 31
4.2.2 Macro AutoHotkey 33
4.2.3 Dashboard 33
4.3 Armazenamento 33
4.3.1 Config.cfg 33
4.3.2 Data.json 34

5 PRODUTO 36
5.1 Instalação 36
5.2 Execução 36
5.3 Comandos 36
5.3.1 Espectador 36
5.3.2 Inscrito 37
5.3.3 Streamer 37
6 ENKIBOT 38
7 CONCLUSÃO E TRABALHOS FUTUROS 44
REFERÊNCIAS 45

1 INTRODUÇÃO

Ao longo do anos, plataformas capazes de permitir que qualquer pessoa possa ter seu próprio canal de tv como, por exemplo, Youtube e Twitch, vêm crescendo em popularidade. No primeiro trimestre de 2020, foram assistidas mais 3.1 bilhões de horas na plataforma da Twitch, e esse número dobrou no mesmo período de 2021, registrando mais de 6.3 bilhões de horas assistidas (FINGAS, 2021). Em janeiro de 2021, o *streamer* David Martínez, bateu o recorde de maior número de espectadores simultâneos em uma mesma *live* da Twitch, com mais de 2.4 milhões de espectadores (SUGGITT, 2021). Entretanto, apesar destes números, essas plataformas digitais disponibilizam poucas ferramentas de apoio para os criadores de conteúdo, especialmente, no que tange o controle de inscritos, gerenciamento de solicitações dos espectadores, etc. Em sua maioria, estas ferramentas de apoio aos *streamers* são desenvolvidas por terceiros, geralmente *chatbots*, como, por exemplo, o Nightbot[1]. Neste sentido, abre-se um leque de possibilidades quando o assunto é a construção de aplicações de auxílio para os criadores de conteúdo, tendo como objetivo o gerenciamento do canal como um todo e, principalmente, controlar a interação do *streamer* com seus espectadores durante as transmissões ao vivo. Atualmente, a interação proporcionada pelos chats existentes nestas plataformas é insuficiente para a maioria dos espectadores e para um grande número de *Streamers*. Assim, torna-se necessário o desenvolvimento de novas ferramentas capazes de estender as funcionalidades básicas dos chat e do canal como um todo, elevando tanto a interação *streamer*-espectador como espectador-espectador para um outro nível (SJöBLOM; HAMARI, 2017).

No âmbito dos canais que abordam o conteúdo de games, o principal desafio é gerenciar a ordem das solicitações dos espectadores que, frequentemente, requisitam a análise de sua jogabilidade pelo *streamer*, mediante a visualização de algum *replay* (partidas previamente gravadas). Assim, além de salvar o *username* e o código do *replay* de quem solicitou tal feedback, estas ferramentas de apoio devem ordenar as solicitações de seus espectadores ao ponto de permitir ao *streamer* atendê-las na mesma sequência de que foram realizadas, ou seja, preservar a ordem cronológica de solicitação ou, até mesmo, atender em um primeiro momento solicitações dos inscritos do canal perante às dos demais espectadores. Atualmente, existem algumas ferramentas que tentam sanar tal problema, porém com código fechado (proprietário) como é o caso do *Warp World Multi-Queue* ou funcionalidades pagas como ocorre no *Play with Viewers*. Desta forma, este trabalho de conclusão de curso propõe a construção de um *chatbot open-source* para o controle de filas em canais de *streamming*, denominado Enkibot, capaz de gerenciar as solicitações dos espectadores através de comandos personalizáveis disparados pelos próprios espectadores durante as *lives* nos chats da Twitch. Além de estender as funcionalidades do

[1]Disponível em: https://nightbot.tv/

chat, o Enkibot possui um *dashboard* gráfico para o gerenciamento de filas que permite ao *streamer* manipular as listas do seu canal da Twitch sem nenhum conhecimento prévio de programação, ou seja, criar novas listas, excluir as solicitações já atendidas e inserir novos pedidos de análise somente somente com alguns cliques no mouse. O Enkibot foi desenvolvido com a linguagem Javascript sobre a plataforma Node.js com a ajuda do framework React e utilizando JSON para armazenamento de dados, pois estas tecnologias auxiliam a construção de aplicações *open-source* multiplataforma.

1.1 Objetivos

O objetivo geral do trabalho é construir um *chatbot open-source* que proporcione ao *streamer* da Twitch e seus espectadores um ambiente amigável de gerenciamento de filas e que, ao mesmo tempo, preserve a ordem das solicitações. Além disso, por ser *open-source*, o Enkibot permite a possibilidade da adição de novas funcionalidades por outros desenvolvedores. Como objetivos específicos, tem-se:

- Estudar tecnologias de desenvolvimento que possibilitem a criação da ferramenta proposta;
- Desenvolver um ambiente intuitivo e interativo, disponibilizado de forma gratuita (*open-source*);
- Criar um ambiente gráfico (*dashboard*) que permita o gerenciamento das filas sem exigir conhecimento prévio de qualquer linguagem de programação por parte do streamer;
- Criar um protótipo que utilize as principais estratégias já conhecidas de gerenciamento de filas de espectadores em serviços de *streaming*;
- Propor um estudo de caso onde o foco é gerenciar as solicitações dos espectadores por revisão de seus *replays* em canais sobre games da Twitch;

1.2 Organização do Texto

Este trabalho foi dividido em capítulos, organizados da seguinte forma:

- No Capítulo 2 é apresentada a fundamentação teórica;
- O Capítulo 3 descreve trabalhos que se assemelham, de alguma forma, ao trabalho proposto, baseado na análise descritiva;
- O Capítulo 4 expõe a etapa de projeto do *chatbot* proposto através de diagramas de caso de uso e de arquitetura, assim como as ferramentas e tecnologias utilizadas;
- O Capítulo 5 apresenta as funcionalidades do aplicativo proposto;
- O Capítulo 6 apresenta um estudo de caso exemplificando o uso das funcionalidades do protótipo, tanto pela visão do *streamer* quanto pela visão do espectador;
- O Capítulo 7 apresenta as conclusões proporcionadas pelo desenvolvimento deste projeto, assim como trabalhos futuros que possam contribuir para o aperfeiçoamento do projeto proposto.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo são abordados os conceitos essenciais que compõem o desenvolvimento deste trabalho. Inicialmente, são definidos termos como **bot**, **chatbot** e **ferramentas de apoio**. Nas seções seguintes, o Capítulo apresenta um breve apanhado das **formas de monetização** mais comuns encontradas nas principais plataformas de *streaming*. Posteriormente, aspectos relacionados às **ferramentas e tecnologias** utilizadas na construção do Enkibot são expostos nas seções seguintes do capítulo.

2.1 Bot

Bot, uma abreviação de *robot* (robô em português), é tipo de software desenvolvido com o intuito de emular ações humanas na Internet, de forma automatizada. (PATEL, 2021). Podem ser divididos em "bons *bots*"e "maus *bots*"dependendo de como são utilizados. Maus *bots*, por exemplo, podem ser utilizados para disseminar vírus ou "trojans"[1], no entanto, bons *bots* são utilizados para auxiliar na automatização de tarefas (DUNHAM KEN; MELNICK, 2009).

Estima-se que metade dos usuários da Internet são *bots* (SUCHACKA; IWAńSKI, 2020). Um exemplo de *bots* da Internet são os rastreadores da rede, os chamados *web crawlers*. *Web crawlers* são sistemas utilizados para realizar o download em massa de páginas da web. Alguns de seus usos são a mineração de dados, em que páginas são analisadas a fim de coletar dados específicos ou até mesmo *web archiving*, que é o ato de arquivar e preservar páginas inteiras (OLSTON; NAJORK, 2010).

2.1.1 Chatbot

A nomenclatura *chatbot* vem da junção de *chat*, em português "conversa", com *bot*. A definição de *chatbot* é um *bot* que recebe comandos de um usuário e gera respostas inteligentes de volta ao usuário através de uma interface de texto (KHAN; DAS, 2017). *Chatbots* possuem métodos para entender solicitações e obter dados de outros softwares para responder essas solicitações (RAJ, 2019).

Além disso, *chatbots* reduzem custos operacionais, mão de obra e tempo, pois melhoram a eficácia e produtividade em setores que lidam com atendimento ao público. Enquanto um funcionário pode atender um cliente por vez, *chatbots* são capazes de interagir com uma quantidade quase ilimitada de pessoas (PAZOS, 2018).

[1] Em português, cavalo de Troia, software enganoso que pode esconder código malicioso.

2.2 Streaming

Streaming é a tecnologia que permite o envio de áudio e vídeo via transferência de dados, através de um dispositivo conectado à Internet. Este envio de dados ocorre através de um fluxo contínuo, ao invés de um download único (SANTANA, 2019). Exemplos comuns de *streaming* são Netflix[2] e Spotify[3], serviços de streaming de vídeo e áudio, respectivamente.

Dentro da tecnologia de *streaming*, existe uma outra modalidade, denominada *live streaming*, que permite a transmissão de dados ao vivo. Conforme os dados são gravados, eles são enviados a todos os dispositivos conectados na *live* (POTENZA, 2017). Essas transmissões podem ser, por exemplo, de eventos ao vivo, vídeo-aulas ou conferências (SANTANA, 2019). Nas *live streamings* que tratam sobre *games*, geralmente, o *streamer* joga um determinado título com a finalidade de reter a atenção da audiência ao vivo e prospectar novos membros do canal, contratos de publicidade e, até mesmo, potenciais compradores deste mesmo jogo. Além disso, um jogo pode se tornar popular devido ao número de *streamers* que venham a fazer *live streaming* deste jogo, fazendo com esta modalidade (*streaming* de games) seja uma forte fonte de divulgação para as empresas desenvolvedoras de *games* (JOHNSON; WOODCOCK, 2018).

2.2.1 Plataformas de *Streaming* de *Games*

Plataformas de *streaming* de *games* são utilizadas por jogadores do mundo inteiro. Os usuários podem utilizar estes sites para compartilhar seus jogos ao vivo, receber feedback, desenvolver uma comunidade e ganhar dinheiro para jogar. (ALVES, 2021).

Plataformas de *live streaming* permitem que espectadores possam conversar com *streamers* e outros espectadores através de um chat, recompensar *streamers* e apoiá-los com assinaturas mensais (LI; WANG; LIU, 2020).

ALVES (2021) cita Twitch e YouTube Gaming, a plataforma do YouTube para *live streaming*, como as melhores plataformas de transmissão de jogos online. Uma das vantagens de começar um canal de *streaming* na Twitch é que você pode utilizar um link para doação para o seu canal, desde o início, mas com o YouTube você precisa ser um Parceiro (KINAST, 2021).

Para se tornar um Parceiro do YouTube, você necessita de 4000 horas de video durante 12 meses e mais de 1000 inscritos, o que pode demorar muito para um canal que está começando. Já na Twitch para se tornar um Parceiro, você precisa de 500 minutos de *streaming* ao longo de 7 dias e ter 50 seguidores e 3 espectadores simultâneos em média (KINAST, 2021).

2.2.2 Streamer e Audiência

Streamer é o indivíduo que realiza uma *live streaming* para espectadores, podendo transmitir, por exemplo, eventos esportivos, análise de produtos como forma de marketing ou *games* em uma plataforma com a Twitch (CHEN et al., 2020). No caso específico dos *streaming* de *games*, as transmissões ao vivo são criadas com intuito de elucidar a habilidade dos *streamers* em um determinado jogo, disponibilizar dicas aos espectadores ou, até mesmo, somente para conversar sobre assuntos variados enquanto estão jogando (LI; WANG; LIU, 2020). Por outro lado, a audiência representa todos os espectadores que consomem o conteúdo produzido por um *streamer*. Geralmente, a audiência interage

[2]Definição em: https://help.netflix.com/pt/node/412

[3]Definição em: https://www.spotify.com/br/about-us/contact/

com o *streamer* através do chat da plataforma utilizada pela transmissão (LI; WANG; LIU, 2020).

2.3 Monetização

Monetização é a forma de receber compensação monetária pela transmissão do canal. Muitos *streamers* tratam o *live streaming* apenas como um passatempo ou uma forma de obter uma interação social, porém, para alguns criadores de conteúdo o *streaming* tornou-se o seu trabalho diário (BOLLATI, 2020).

2.3.1 Formas de Monetização

Os *streamers* podem receber apoio através de patrocínio por empresas do meio dos *games*, para anunciar produtos ou serviços. Estima-se que o *streamer* "Ninja", tenha recebido 1 milhão de dólares para fazer *stream* do jogo "Apex Legends"em 2019, na Twitch (PANCHADAR, 2019).

Os sites de *streaming* disponibilizam ferramentas para que os *streamers* possam ser remunerados dentro da própria plataforma, sendo a principal delas uma forma de mensalidade paga aos *streamers* pelos espectadores. Além disso, os espectadores podem comprar métodos para terem suas mensagens no chat destacadas, os *Super Chats* no caso do YouTube ou utilizando *Bits* na plataforma da Twitch. Pode-se também, caso configurado, doar diretamente ao *streamer* através de links de plataformas de transferência monetária, como PicPay[4]. Algumas das formas de monetização de um canal estão listadas abaixo:

- **Doação direta**: Podem ser configurados links no canal para doação direto ao *streamer*.
- **Super Chat/Super Stickers**: São mensagens ou figuras compradas que dão destaque ao comentário do espectador/comprador - fixadas, temporariamente, no topo do chat do canal do YouTube (YOUTUBE, 2021).
- **Bits**: É uma "moeda"adquirida diretamente no site da Twitch e pode ser utilizada no chat de um *streamer*. Fica ao critério do usuário definir a quantidade de bits que serão doados ao *streamer* por meio do comando *"cheer"* (TWITCH, 2021a).
- **Assinaturas e Inscrições**: As assinaturas do YouTube e inscrições da Twitch são modelos de pagamentos mensais, com valores fixos que podem variar dependendo da plataforma de *streaming* utilizada e país. Usuários que venham aderir a alguma modalidade de assinatura ou inscrição recebem vantagens no canal como, por exemplo, figurinhas exclusivas, prioridade em promoções e, até mesmo, acesso a conteúdo exclusivo.
- **Propagandas**: Os espectadores podem assistir propagandas em vídeo que são transmitidas automaticamente pela plataforma ou agendadas pelo *streamer*.

2.4 Ferramentas de Apoio para Streamers

Ferramentas de Apoio são instrumentos auxiliadores para facilitar a execução de tarefas muitas vezes repetitivas, que demandariam muito tempo ou, simplesmente, seriam

[4]https://www.picpay.com/site

inviáveis a um usuário comum. Estas ferramentas de apoio são, geralmente, disponibilizadas como uma extensão de um navegador, um *plugin*[5] de um aplicativo ou como um *bot* (DARE DROP, 2020).

Ao navegar por páginas da Internet, o usuário pode se deparar com uma infinidade de propagandas. Uma das alternativas para evitar a exposição à quantidade excessiva de anúncios é a instalação do AdBlock[6]. O AdBlock é uma extensão para navegadores *web* que impede o carregamento de certos anúncios na página a partir de um conjunto de regras criado pelo usuário (YEOW, 2005). Além disso, é disponibilizado, por padrão, um filtro criado a partir de uma lista de anúncios conhecidos chamado EasyList[7].

Existem ferramentas de apoio criadas especificamente para *streamers*, utilizadas tanto para a criação e execução da transmissão, quanto para o gerenciamento do chat. Como a interação entre *streamer* e espectador é feita através do chat das plataformas de *streaming*, é interessante a utilização de *chatbots* para aperfeiçoar a utilização deste canal de comunicação entre os usuários. Um *chatbot* conhecido é o Nightbot, disponível tanto para o YouTube quanto para a Twitch. O Nightbot oferece muitas funcionalidades para os *streamers* como a criação de sorteios para os espectadores, proteção contra *spam*[8] ou links maliciosos que espectadores podem digitar no chat (MORRIS, 2019). A Twitch, por exemplo, disponibiliza algumas destas ferramentas de apoio, enquanto que *chatbots* como o Nightbot, são disponibilizados por terceiros.

O artigo de LI; WANG; LIU (2020) comenta que meios de incentivar a interação entre *streamer* e espectadores são importantes, pois fazem que o espectador fique mais tempo assistindo a transmissão do canal e, consequentemente, assista mais propagandas ou, até mesmo, esteja compelido a apoiar o canal com inscrição ou doações. Essa atividades, inclusive, podem ajudar a criar um senso de comunidade entre *streamer* e seus espectadores. Vale ressaltar ainda que o gerenciamento de filas pode ser aproveitado fora do escopo de *streaming* de *games*, sendo utilizado pelos *streamers* para criar uma fila com, por exemplo, uma lista de indicações de vídeos no YouTube a serem assistidos nas *lives*, juntamente com seus espectadores.

Uma das ferramentas de interação com os espectadores disponibilizas pela Twitch, é a criação de enquetes com participação diretamente no chat. A Figura 2.1 ilustra as opções para criação de uma enquete. Nesta enquete, é possível escrever 1 pergunta de até 60 caracteres com até 5 opções de resposta, habilitar a votação com *bits*, multiplicar o valor dos votos por 2 e permitir que somente os inscritos votem. Além disso, é obrigatório definir o período de tempo em que a enquete estará aberta à votações.

Pontos do Canal é outra ferramenta de apoio que pode ser usada pelo *streamer*. Este programa de pontos personalizável está disponível somente para Parceiros e Afiliados e permite aos *streamers* presentear seus espectadores com um conjunto de vantagens, inclusive com uma amostra dos benefícios reservados somente aos membros inscritos (TWITCH, 2021b). É possível adquirir Pontos do Canal sem custo monetário. A cada 5 minutos de transmissão assistidos pelo espectador, ele recebe no mínimo 10 pontos. O *streamer* pode definir recompensas a serem resgatadas pelos espectadores, utilizando o painel de controle do criador.

[5]Módulo ou extensão para adicionar funções à programas existentes
[6]Disponível em: https://getadblock.com/pt_BR/
[7]Disponível em: https://easylist.to/
[8]Mensagens repetidas em massa

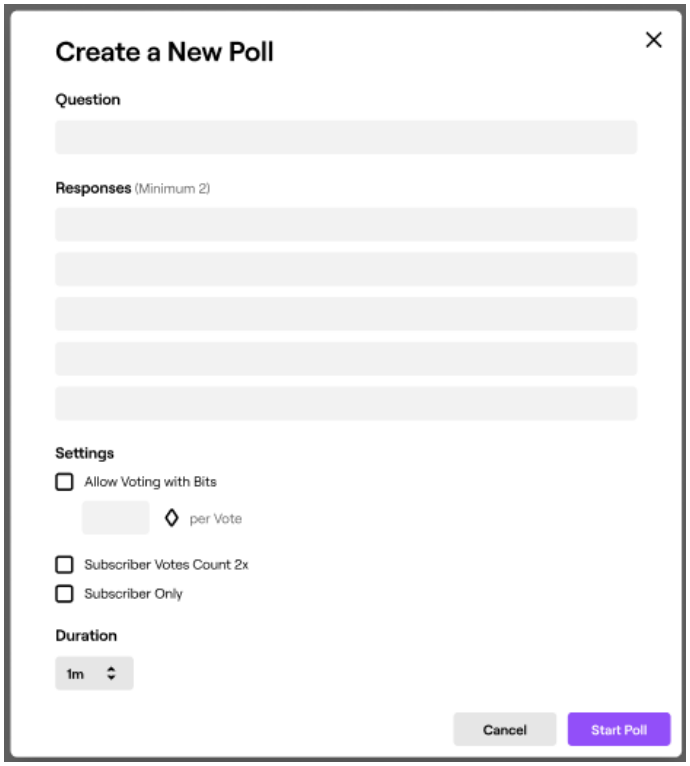

Figura 2.1: Menu de criação de enquetes na Twitch.

Como ilustrado pela Figura 2.2, algumas opções vêm pré-configuradas por padrão, dentre elas, habilitar aleatoriamente, ou por escolha, um emote[9] exclusivo de inscritos por 24 horas ou destacar uma mensagem no chat. O *streamer* pode também criar novas recompensas para o seu canal, contanto que estejam dentro da Política de Uso[10] do Pontos de Canal.

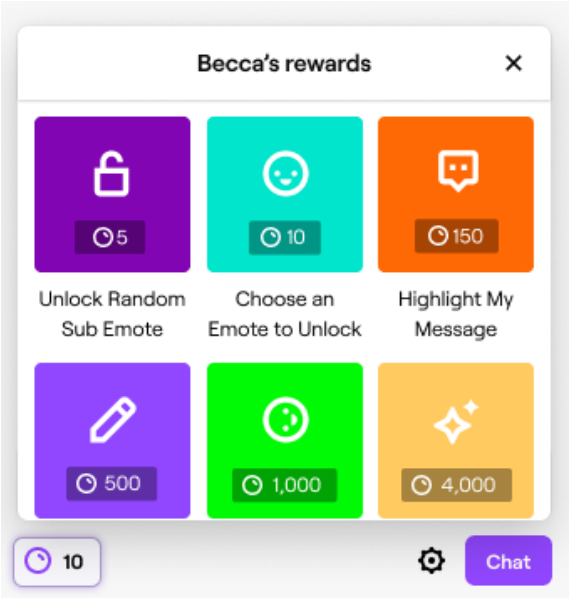

Figura 2.2: Menu dos Pontos de Canal na Twitch.

[9]Descrição em: https://www.twitch.tv/creatorcamp/pt-br/learn-the-basics/emotes/
[10]Disponível em: https://www.twitch.tv/p/pt-br/legal/channel-points-acceptable-use-policy/

2.5 Tecnologias de Desenvolvimento

Nesta seção serão mencionadas as tecnologias utilizadas neste trabalho, bem como explicar o motivo da escolha das mesmas.

2.5.1 Node.js

Node.js é um ambiente de execução JavaScript, *open-source* e *cross-platform* (NODEJS, 2021). Foi escolhido Node.js para o projeto devido à fácil implementação com a biblioteca *tmi.js*, para integração de um *client* com o chat da Twitch. Outro ponto é o projeto exigir um número elevado de leituras em um arquivo JSON para salvar o estado da fila de usuários e o Node.js é desenvolvido numa arquitetura não-bloqueante de *threads* e exibe uma performance boa em relação ao uso de memória (PEREIRA, 2014).

2.5.2 tmi.js

O *tmi.js* é uma biblioteca Javascript para a interface de mensageria da Twitch (FOSTER, 2021). Esta biblioteca foi escolhida por conta de sua simplicidade e facilidade de configuração para integração de um *chatbot* com o chat da Twitch. Com a *tmi.js* é possível enviar mensagens para o chat da transmissão, assim como identificar as mensagens de outros usuários, possibilitando a utilização das mesmas para ativar comandos definidos no *chatbot*.

2.5.3 React

React é uma biblioteca JavaScript para criar interfaces de usuário em páginas *web* (REACT, 2021). Diferente de *frameworks* como AngularJS[11], React utiliza JavaScript ao invés de HTML para construir essas interfaces de usuário (HEMEL, 2013). Foi utilizado React, pois um componente pode receber os dados de entrada, como, por exemplo, de um arquivo JSON, fazendo com que o *dashboard* seja atualizado toda vez que exista uma alteração no arquivo automaticamente, sem interação do usuário.

2.5.4 AutoHotkey

AutoHotkey é uma linguagem *open-source* de desenvolvimento de *scripts* para Windows (AUTOHOTKEY, 2021). A escolha de sua utilização é devido ao fato de poder enviar a simulação de batida de teclas a outras janelas do sistema operacional, permitindo que os dados do *chatbot* sejam transmitidos a jogos que estão sendo executados na máquina do usuário.

[11] Definição em: https://angularjs.org/

3 SISTEMAS EXISTENTES

Este Capítulo analisa alguns softwares existentes para facilitação no gerenciamento de filas de espectadores e inscritos de um canal na Twitch. Além disso, o Capítulo apresenta as funcionalidades de cada software e suas diferenças no intuito de apontar a abordagem utilizada por cada aplicação para a resolução do problema de controle de filas dos espectadores.

3.1 Twitch Queue Bot

Twitch Queue Bot é um software desenvolvido em Java para gerenciar filas em um canal da Twitch (CHURCHILL, 2020a). As filas auxiliam os *streamers*, pois definem uma ordem de interação do *streamer* com os seus espectadores, ou seja, qual espectador terá seu comentário ou pedido atendido antes dos demais que estão em uma mesma fila. Além disso, na Twitch Queue Bot, o *streamer* pode organizar seus espectadores em mais de uma fila, sendo possível criar filas exclusivas para os inscritos do seu canal. Este gerenciamento é feito através de um sistema de um *bot* configurado por uma interface escrita em JavaFX[1], que se conecta ao chat do canal da Twitch do *streamer*. A Figura 3.1 ilustra esta interface de configuração que estabelece a conexão do *bot* com o canal da Twitch. Nesta mesma tela o *streamer* pode customizar outros parâmetros como, por exemplo, um identificador de *replay*, o nome da conta do espectador dentro do jogo ou, simplesmente, uma mensagem ao *streamer*.

O usuário pode acompanhar em tempo real todas as informações relacionadas às filas como a lista de espectadores e a prioridade dos inscritos (Figura 3.2). Além disso, há o histórico de todos os participantes, como, por exemplo, o número de vezes em que o mesmo espectador já esteve na fila ou se deixou alguma mensagem ainda não lida.

[1]Definição em: https://openjfx.io/

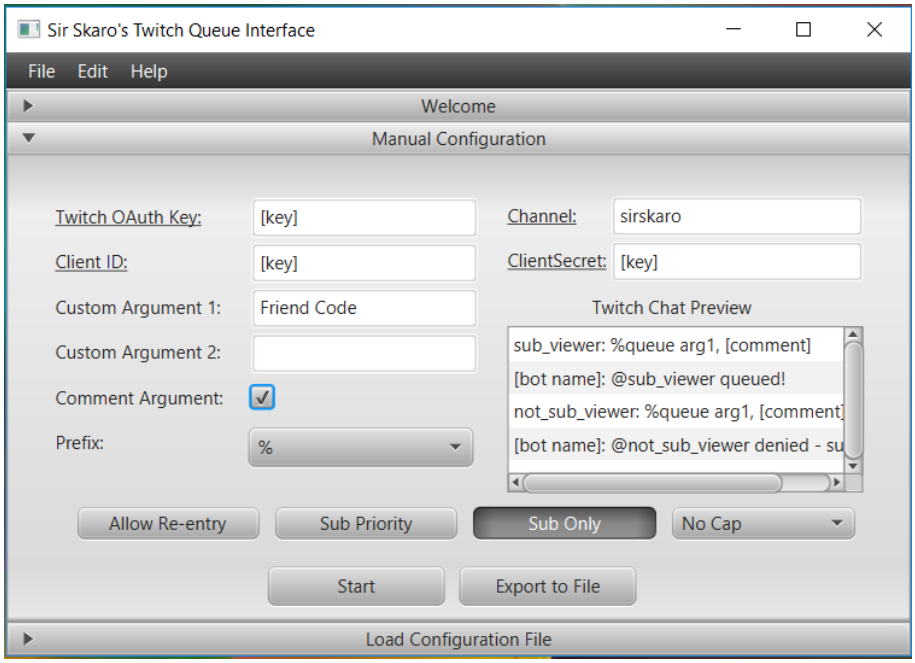

Figura 3.1: Twitch Queue Bot: Interface de Configuração.

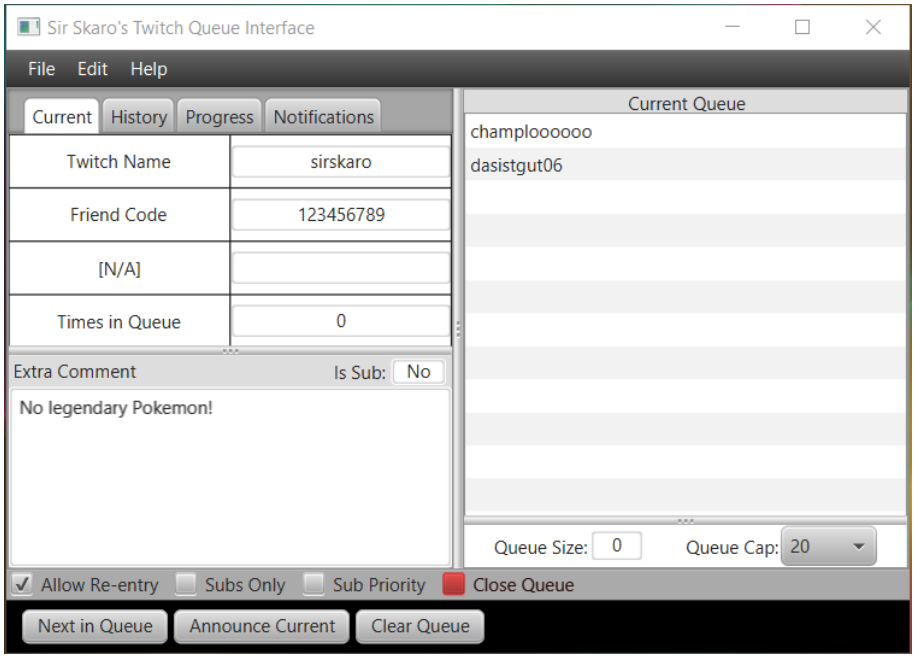

Figura 3.2: Twitch Queue Bot: Interface de Estado da Fila.

3.2 Play with Viewers

O Play With Viewers é uma extensão construída para a Twitch onde o uso, configuração e instalação são realizadas, internamente, no próprio canal do *streamer* na Twitch. A extensão disponibiliza informações a respeito das filas tanto para o *streamer* quanto para os espectadores do canal, conforme ilustrado pela Figura 3.3. Além disso, a ferramenta permite também, internamente, uma interação via chat de maneira *drag-and-drop*.

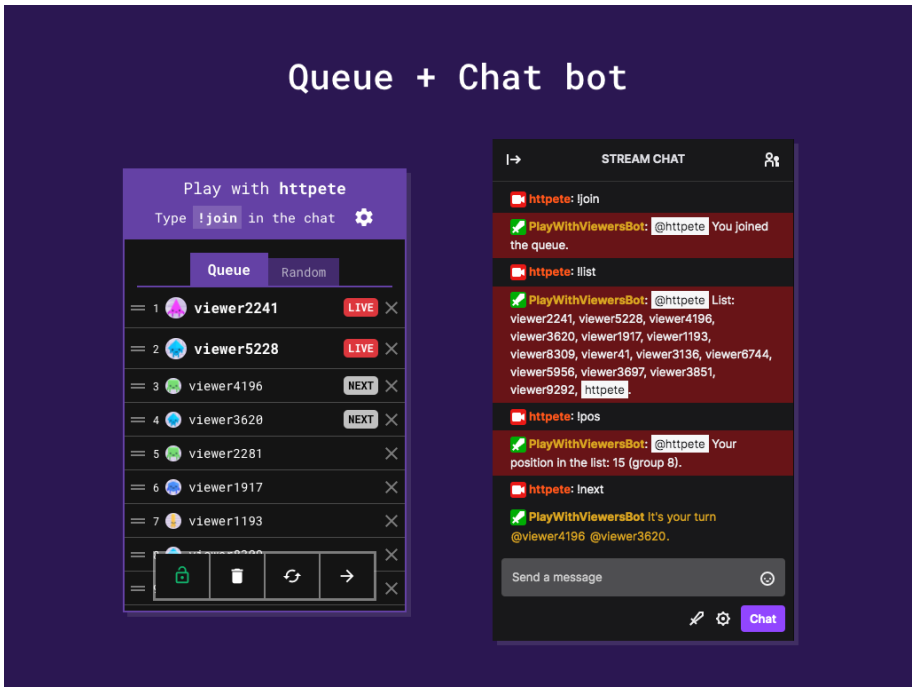

Figura 3.3: Play With Viewers: Interface para o controle de filas e chat do canal da Twitch.

Os principais recursos da ferramenta são o modo aleatório (que permite a escolha aleatória de espectadores da fila) e a utilização de *bits*[2] (moeda paga dentro da própria Twitch, para obter prioridade na fila) por parte dos espectadores com o propósito de ganhar prioridade na fila e apoiar o canal (GOUTHERAUD, 2020). Além da Twitch, a ferramenta permite a integração ao Discord, onde o *streamer* pode configurar seu canal para receber os mesmo comandos do chat da Twitch e adicionar pessoas à mesma fila da extensão. A Figura 3.4 ilustra o modo aleatório (*Random-Mode*) da extensão Play With Viewers onde é possível chamar um espectador da fila de maneira aleatória e mostrar a possibilidade do uso de *bits*.

O Play With Viewers permite configurar novos comandos para as funções já existentes, como, por exemplo, abrir e fechar fila, participar e sair da fila, assim como configurar uma mensagem personalizada para cada comando, sendo ilustrado pela Figura 3.5.

[2]Definição em: https://www.twitch.tv/creatorcamp/pt-br/get-rewarded/bits-and-subscriptions/

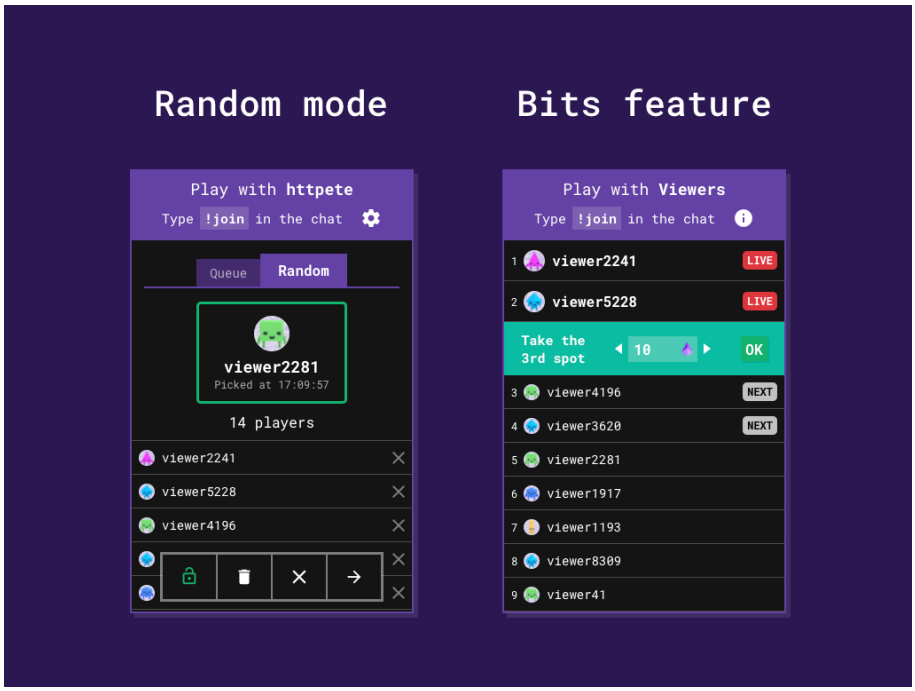

Figura 3.4: Play With Viewers: Modo aleatório e Recurso de *bits*.

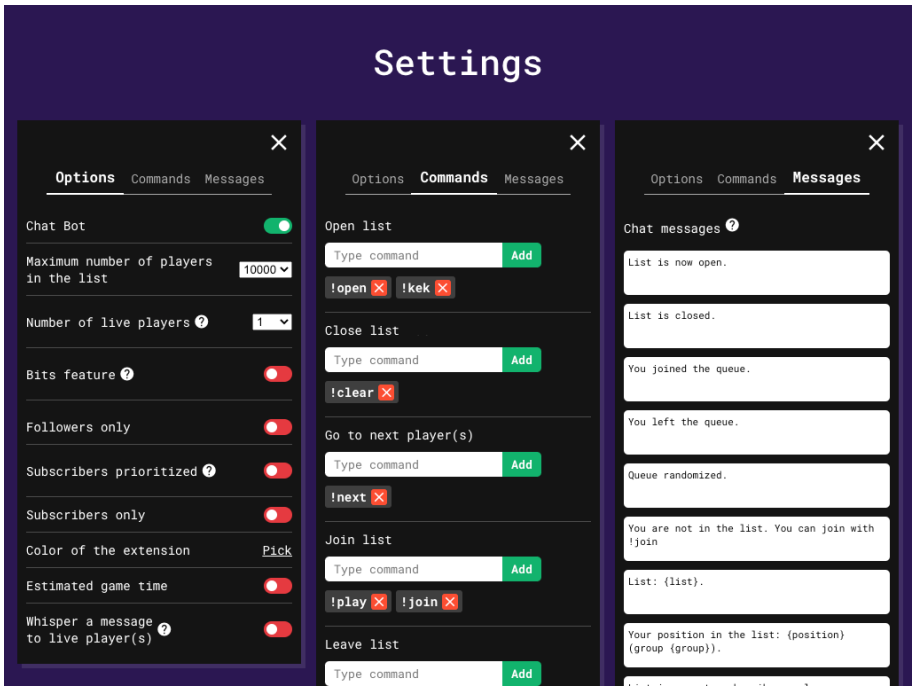

Figura 3.5: Play With Viewers: Telas de Configuração.

A Figura 3.6 ilustra a fila somente para os inscritos em um determinado canal da Twitch. Os valores dos planos de assinatura podem variar de preço de acordo com a modalidade contratada (mensal por US$ 3 ou vitalício por US$ 20 dólares).

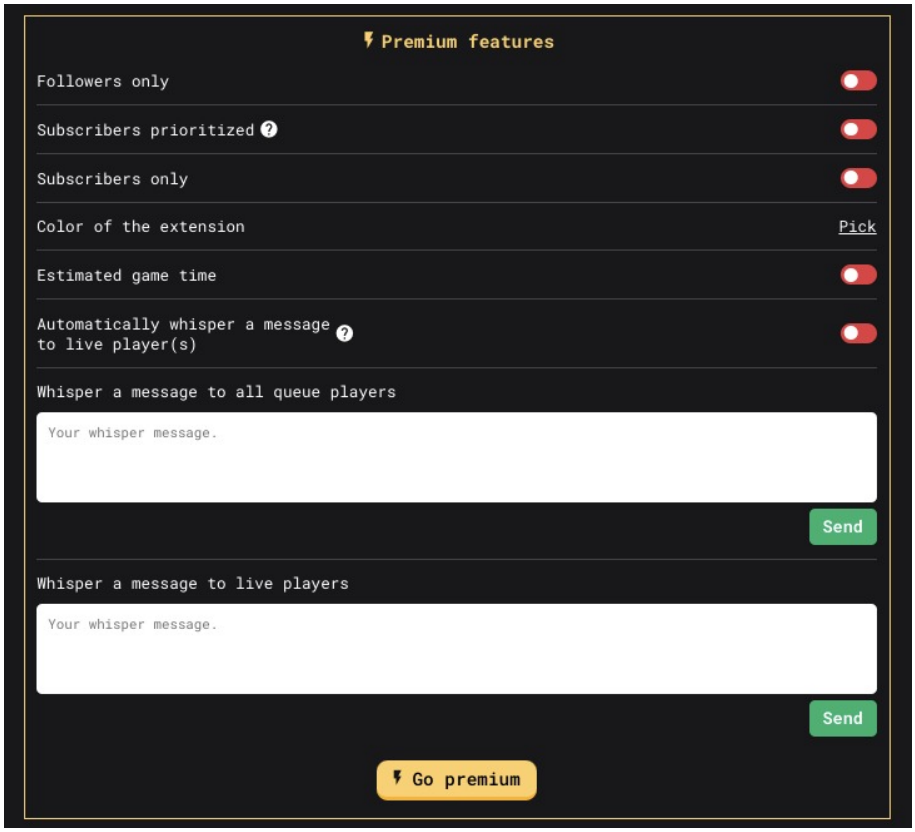

Figura 3.6: Play With Viewers: Recursos Adicionais do Plano Pago (*Premium*).

3.3 Warp World Multi-Queue

Warp World é um site com algumas ferramentas para auxiliar *streamers* em alguns jogos, dentre elas o Warp World Multi-Queue. O Warp World Multi-Queue possibilita o gerenciamento de filas através do próprio site, sendo que o *streamer* pode utilizar tanto sua conta ou a do site como *bot* para receber comandos no chat da Twitch (JAKU, 2020).

A integração é feita em tempo real e toda configuração através de um *dashboard* no site do Warp World, conforme Figuras 3.7 e 3.8. A ferramenta possui opções de configurações comuns a outras ferramentas, como abrir e fechar filas, limitação de número de participantes, filas exclusivas para seguidores ou inscritos, integração com o uso de *bits* e participação da fila via Discord. Também permite salvar múltiplas filas, permitindo ao *streamer* utilizá-las em diferentes ocasiões.

Um diferencial é a configuração do *Warp Bar*, um *widget* para mostrar informações da fila com integração ao OBS ou xSplit[3]. É possível configurar tanto a aparência da *Warp Bar* como, por exemplo, cor do preenchimento, tamanho e cor da borda, quanto as informações que serão exibidas (Figuras 3.9 e 3.10).

[3]OBS Studio e XSplit são softwares utilizados para a gravação de desktops e streaming de jogos.

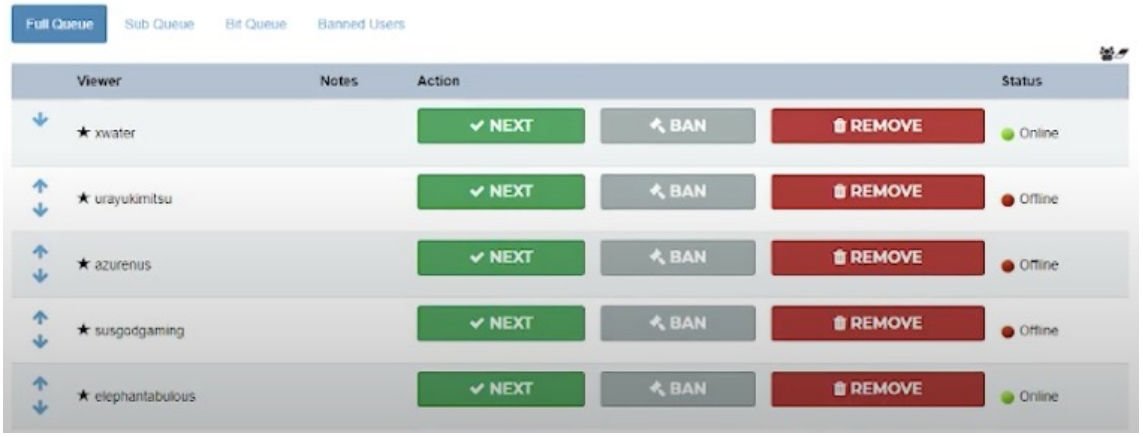

Figura 3.7: Warp World: *Dashboard* da fila de espectadores.

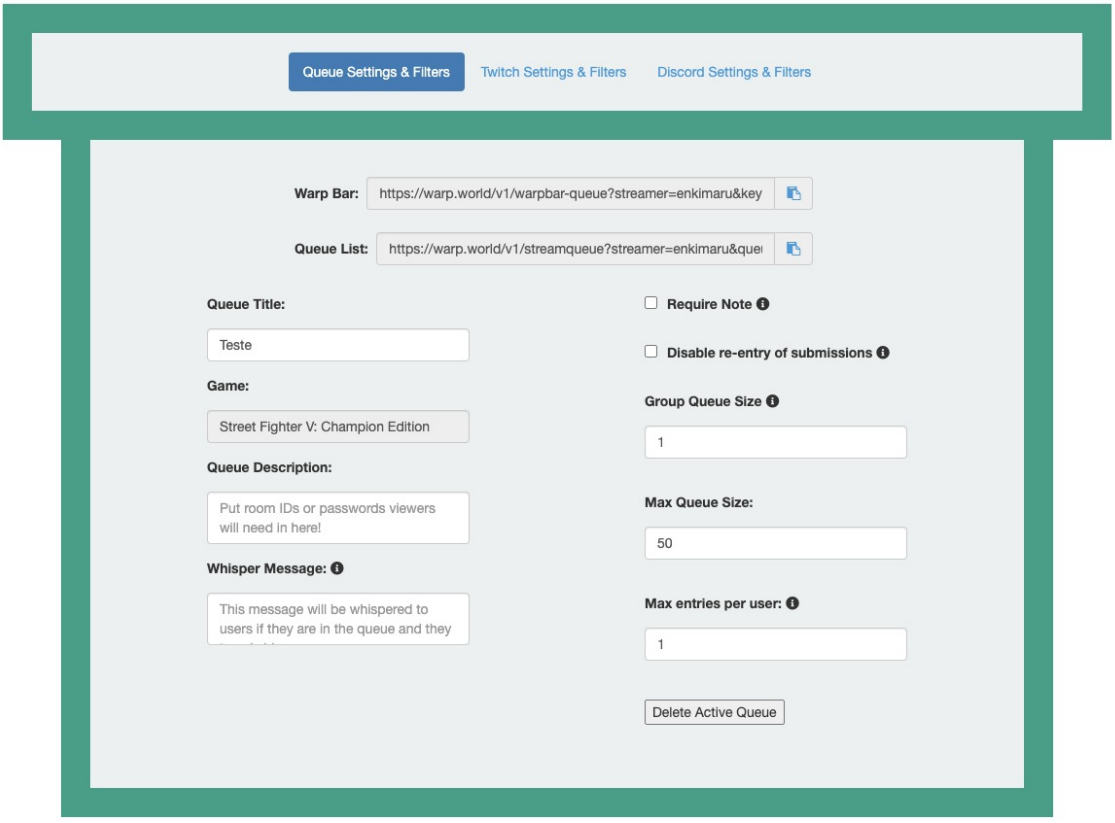

Figura 3.8: Warp World: Tela de configuração da fila.

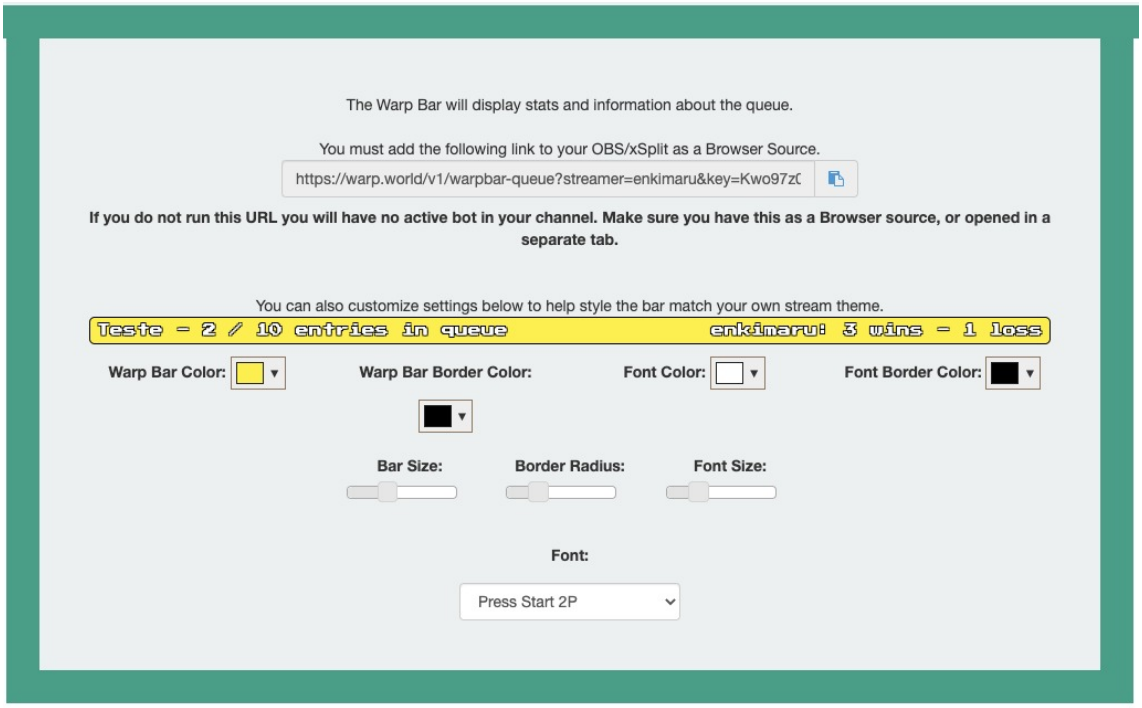

Figura 3.9: Warp World: Página de configuração do visual da *Warp Bar*.

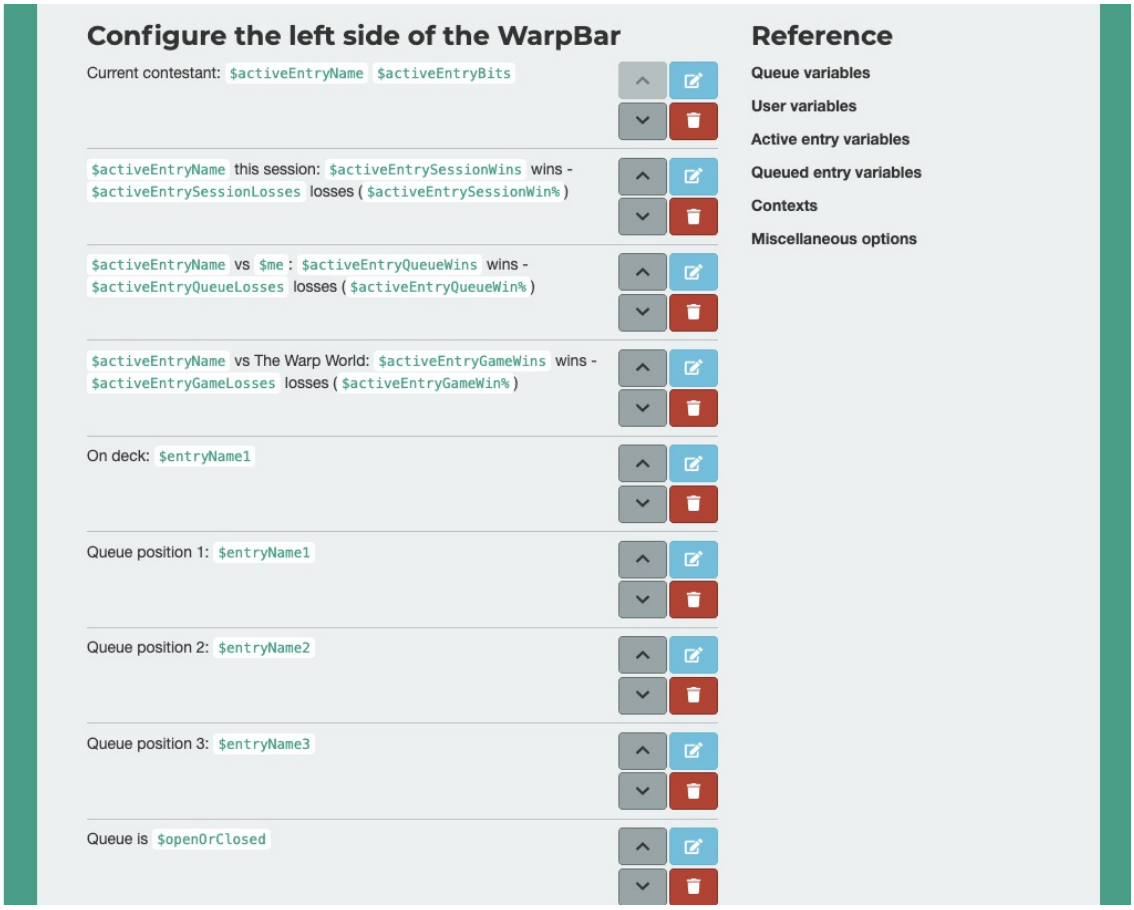

Figura 3.10: Warp World: Página de configuração da *Warp Bar*.

3.4 Comparativo entre os Sistemas Existentes

Esta Seção faz um comparativo a respeito das funcionalidades dos sistemas pesquisados, evidenciando suas diferenças e semelhanças. A Tabela 3.1 apresenta as principais características de cada ferramenta e compara tais características com o *chatbot* Enkibot proposto por este trabalho.

O Twitch Queue Bot, por ser um software *open-source*, permite que novas funcionalidades sejam implementadas por outros desenvolvedores de software. Entretanto, é um projeto que está há algum tempo sem receber nenhuma melhoria (*upgrade*) e em virtude das recentes atualizações do Java, acaba exigindo de seus usuários algum conhecimento técnico prévio, principalmente, na etapa de instalação - o que inviabiliza sua utilização para *streamers* não-desenvolvedores (CHURCHILL, 2020b). Por sua vez, o Play With Viewers possui uma vantagem neste quesito, pois é uma extensão que pode ser utilizada diretamente na Twitch, sem nenhum *download* e instalação prévios ou, até mesmo, a necessidade da criação de uma nova uma conta em algum serviço externo. Além disso, a fila pode ser observada tanto pelo *streamer* quanto pelos espectadores na mesma página em que acontece a *stream*, não havendo a necessidade do uso de outra janela. Caso o *streamer* deseje utilizar o sistema de *bits*, sofrerá um desconto de 20% no valor da moeda, porcentagem esta direcionada ao autor da ferramenta[4]. Vale ressaltar que a utilização de *bits* para criar a promessa de jogar com um usuário específico em um jogo viola a política de uso de *bits* da Twitch (TWITCH, 2021c). Além disso, possui no plano pago funcionalidades exclusivas que em outras ferramentas são consideradas como básicas e/ou gratuitas, como, por exemplo, prioridade de inscritos ou fila exclusiva para um tipo de expectador. Além disso, no Play With Viewers não permite ao usuário trocar sua conta da Twitch, ou seja, se o *streamer* estiver com sua conta desabilitada por algum motivo, não poderá cadastrar uma nova. Por outro lado, o Warp World Multi-Queue permite a troca da conta atual por qualquer outra conta da Twitch, mas, da mesma forma que o Play With Viewers, não está localmente instalada na máquina do *streamer* e poderá sofrer momentos de indisponibilidade caso o servidor esteja *off-line*.

Tabela 3.1: Tabela Comparativa dos Sistemas Existentes.

Nome	**Tipo**	**Open Source**	**Conteúdo Pago**	**Pontos do Canal**	**Integração com Jogos**	**Integração com Discord**
Twitch Queue Bot	Software	Sim	Não	Não	Não	Não
Play with Viewers	Extensão	Não	Sim	Não	Não	Sim
Warp World Multi-Queue	Website	Não	Sim	Não	Sim	Sim
Enkibot	Software	Sim	Não	Sim	Sim	Não

[4]https://help.twitch.tv/s/article/earning-revenue-from-bits-in-extensions

4 PROJETO

Neste Capítulo será apresentada a arquitetura do Enkibot e as funcionalidades do aplicativo proposto por meio de diagramas UML de caso de uso.

4.1 Diagrama de Casos de Uso

O diagrama UML da Figura 4.1 apresenta os casos de uso do Enkibot organizados por pacotes (**Chatbot**, **Dashboard** e **Macro AutoHotKey**) e por atores (**Streamer**, **Inscrito** e **Espectador**).

4.1.1 Atores

O **Streamer** é o dono do canal e organiza a fila para ter a interação com os seus espectadores. O **Espectador** é o visitante do canal e assiste o conteúdo transmitido pelo **Streamer**, podendo se cadastrar à fila de solicitações por meio de pontos do canal - caso a restrição de pontos esteja configurada previamente. O ator **Inscrito** representa um apoiador do canal que paga mensalmente por uma assinatura, obtendo vantagens como prioridade em filas - sem a necessidade de utilizar pontos. Tanto **Espectador** quanto **Inscrito** só podem utilizar os comandos do Enkibot no *chat*. Por sua vez, o *Streamer* pode ter acesso as filas pelo *dashboard* e ao macro AutoHotKey.

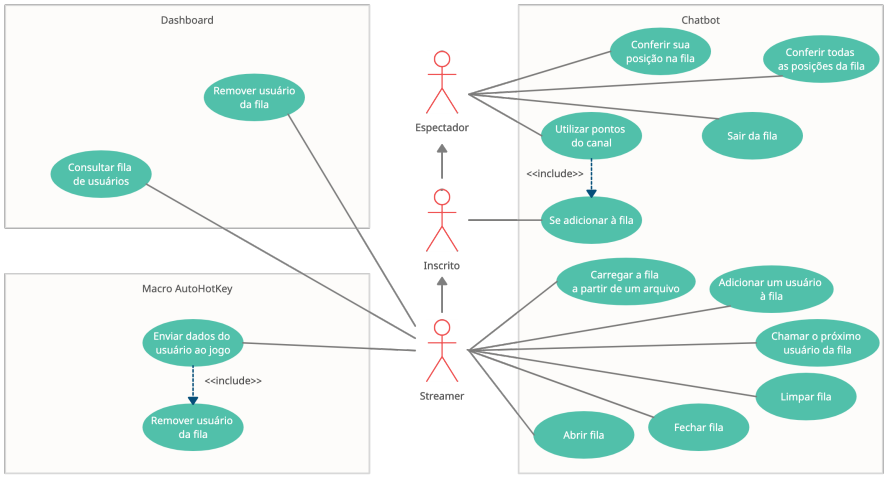

Figura 4.1: Diagrama de Casos de Uso.

4.1.2 Pacotes

Os Pacotes são conjuntos de Casos de Uso separados pra cada aplicação relacionada ao Enkibot. Enquanto que todos os atores podem utilizar o *Chatbot*, apenas o **Streamer** pode utilizar o *Dashboard* e o *Macro Autohotkey*.

4.1.2.1 Chatbot

O **Chatbot** tem como função se conectar ao chat da Twitch e receber comandos, tanto do **Streamer** quanto dos espectadores. Além disso, também manipula os dados recebidos e retorna a resposta ao chat da Twitch. Dentro do pacote **Chatbot**, existe os seguintes caso de uso:

- **Abrir Fila:** Comando para habilitar a adição de novos espectadores à fila.
- **Fechar Fila:** Comando para impedir que novos espectadores entrem na fila, mas não a deleta.
- **Limpar Fila:** Remove todos os usuários da fila.
- **Chamar o próximo usuário da fila:** Operação que remove o primeiro espectador da fila e chama o próximo, notificando o mesmo no chat da Twitch através de uma menção.
- **Adicionar um usuário à fila:** Comando para adicionar um espectador qualquer à fila, utilizado somente pelo *streamer*.
- **Carregar a fila a partir de um arquivo:** Carrega a fila a partir do JSON salvo, caso necessite fazer um backup e queira trocar com o *charbot* estando online. Este comando é sempre carregado toda vez que o *chatbot* é iniciado.
- **Se adicionar à fila:** Operação para que um usuário possa entrar na fila. É necessário que a fila esteja aberta para que o usuário seja adicionado. É possível adicionar parâmetros adicionais ao comando para informar algum dado ao *streamer*, por exemplo, o nome da conta ou alguma mensagem. Através das configurações, é possível limitar o uso do comando somente a espectadores inscritos.
- **Gastar Pontos do Canal:** Caso habilitado, permite ao espectador não-inscrito se juntar a uma fila aberta, utilizando pontos do canal[1] como entrada.
- **Sair da fila:** Permite ao usuário sair da fila.
- **Conferir sua posição na fila:** Comando para o *chatbot* informar ao usuário, no chat da Twitch, qual a sua posição atual na fila.
- **Conferir todas as posições da fila:** Comando que faz com o que o Enkibot informe no chat a lista inteira da fila, organizada pela posição do primeiro ao último espectador.

[1] Definição em: https://help.twitch.tv/s/article/channel-points-guide

4.1.2.2 Macro AutoHotKey

O pacote **Macro AutoHotkey**[2] do diagrama de caso de uso representa um *script* desenvolvido na própria linguagem que permite o envio de dados para qualquer aplicação que rode no Windows. No caso específico do Enkibot, o **Macro AutoHotkey** extrai parâmetros adicionados por usuários que estão em uma das filas e os envia para um determinado jogo (caso seja de interesse do **Streamer**). Além disso, pode também criar um atalho no teclado que remove o usuário do topo da fila.

4.1.2.3 Dashboard

O **Dashboard** é uma tabela exibida em uma página web que roda em paralelo ao **Chatbot**. Tem como principal função exibir em tempo real as informações das filas de forma organizada para o **Streamer**. Como mostrado na Figura 4.2, o **Dashboard** exibe o nome de cada usuário, a mensagem adicional (caso enviada) e se o usuário é inscrito - ou não - no canal. Além disso, o **Dashboard** disponibiliza também um botão que permite remover um usuário específico da fila de acordo com a escolha do **Streamer**.

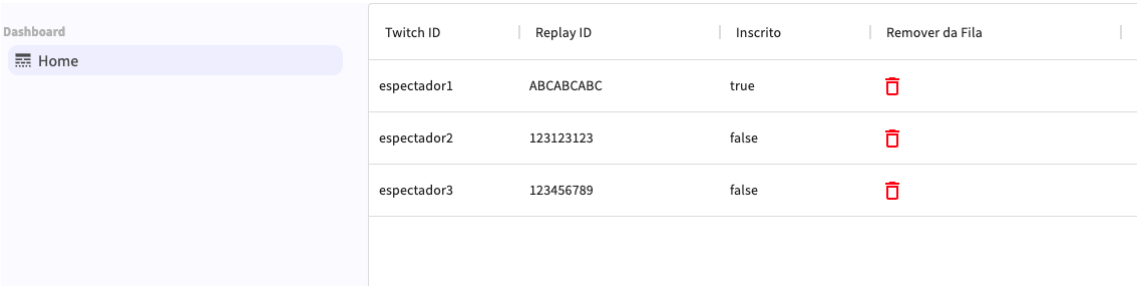

Figura 4.2: Dashboard do Enkibot.

4.2 Arquitetura

A seção de Arquitetura tem o intuito de definir os componentes e seus relacionamentos dentro de cada aplicação que, em conjunto, definem a estrutura geral do Enkibot. Serão definidos e explicados os módulos e métodos que cada aplicação possui.

4.2.1 Chatbot

A arquitetura do *Chatbot* é dividida em 4 módulos principais: App, RoutesReact, Functions e Client, ilustrados pela Figura 4.3.

4.2.1.1 App

Módulo que integra outros 3 sub-módulos. O módulo *App* implementa o *Client* e recebe os comandos do chat da Twitch. Além disso, transcreve as mensagens escritas pelos espectadores ou pelo próprio *Streamer*. O *Client* verifica se o comando começa com **'!'** para então decidir se o comando tem uma palavra-chave válida. Também verifica se o usuário tem permissão para acessar certos comandos, mesmo que a grafia esteja correta. Nenhum método do módulo *App* interage diretamente com a fila, eles implementam a lógica para decidir qual método do módulo *Functions* chamar, baseado nas informações enviadas ao *Client*. O módulo *App* implementa os seguintes métodos:

[2]https://www.autohotkey.com/

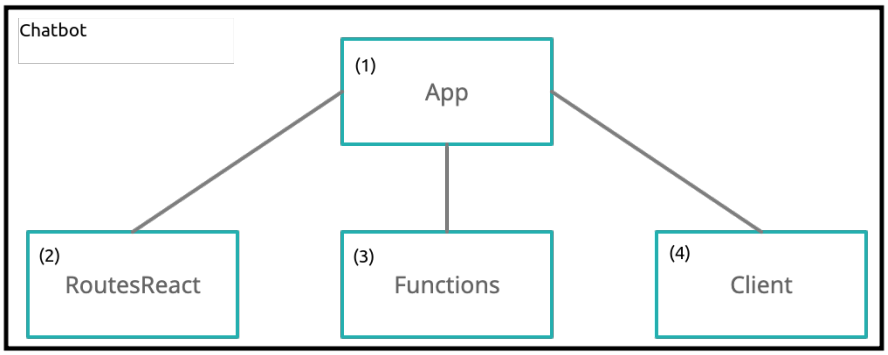

Figura 4.3: Arquitetura do Chatbot.

- **addUserToQueue:** Adiciona um usuário na fila. Caso o usuário que utiliza esse comando seja um *streamer* ou um moderador do canal ele poderá, inclusive, adicionar um usuário qualquer à fila, caso contrário, o usuário poderá adicionar apenas ele mesmo. Se o usuário em questão for um espectador não inscrito no canal, deverá utilizar os pontos do canal para poder utilizar o comando.
- **showQueue:** Exibe todos os usuários inclusos na fila.
- **openQueue:** Abre a fila, habilitando a entrada de novos usuários.
- **closeQueue:** Fecha a fila, mas não a deleta, apenas impedindo a entrada de novos usuários.
- **nextOnQueue:** Deleta o primeiro da fila e chama no chat o próximo usuário.
- **leaveQueue:** Remove o usuário que utilizou o comando, da fila.
- **positionOnQueue:** Exibe no chat a posição do usuário na fila.
- **loadQueue:** Carrega a fila a partir do arquivo JSON especificado no arquivo de configurações.
- **deleteQueue:** Deleta a fila.

Também implementa a lógica para utilizar os pontos do canal para adicionar um usuário na fila.

4.2.1.2 RoutesReact

Implementa o *framework Express*[3] para receber requisições REST da *dashboard*. Gerencia a rota para deletar um usuário da fila e também configura a porta em que o *chatbot* espera essas requisições.

4.2.1.3 Functions

Este módulo lida diretamente com a fila e recebe chamadas do módulos *App* e *RoutesReact*. Implementa as bibliotecas *crypto* para gerar um UUID para cada usuário adicionado da fila e manipular o arquivo JSON, onde será salvo a informação da fila.

[3]Definição em: https://expressjs.com/pt-br/

4.2.1.4 Client

O módulo *Client* necessita das configurações salvas no arquivo *Config.cfg* e da biblioteca *tmi.js*. Realiza a conexão do *chatbot* ao chat da Twitch, enviando usuário e autenticação do *chatbot* e a qual canal deseja se conectar.

4.2.2 Macro AutoHotkey

A aplicação Macro AutoHotkey dispõe de apenas um módulo chamado **Helper.ahk**, que possui apenas dois métodos, **getNextOnQueue** e **sendInfoToCurrentWindow**. O método **getNextOnQueue** remove o primeiro usuário da fila do arquivo *Data.json* e, caso exista, salva em memória a mensagem cadastrada pelo usuário. Já o método **sendInfoToCurrentWindow** envia à janela ativa do sistema operacional a mensagem salva pelo método **getNextOnQueue**.

4.2.3 Dashboard

A arquitetura do **Dashboard** possui um módulo principal **UserList**, que lida com a lógica da aplicação, preparando a tabela com a lista de usuários da fila e também disponibilizando um botão ao lado de cada usuário com o seu identificador permitindo a remoção do mesmo, através de uma requisição REST feita ao *chatbot*.

Através do método **getData**, a **UserList** recebe todos os usuários encontrados no arquivo *Data.json* e utiliza o hook[4] **useEffect** para analisar possíveis mudanças na fila provenientes de outras aplicações. Após os dados serem recebidos com sucesso, a função **setData** é executada e uma nova tabela é gerada com a fila atualizada. Por fim, a função **handleDelete** recebe como um parâmetro o identificador único de caso usuário e, caso disparada, faz uma requisição ao *chatbot* para que o usuário seja removido da fila.

4.3 Armazenamento

O Enkibot não utiliza banco de dados, a persistência de dados é feita em 2 arquivos principais, *Config.cfg* e *Data.json*.

4.3.1 Config.cfg

O arquivo *Config.cfg* contém os parâmetros de configuração que devem ser definidos pelo *streamer* para a correta utilização do *Enkibot*. Dentre estes parâmetros há a definição do nome do canal (*channel*), o nome do bot no chat (*botUsername*), qual será o comando que adiciona um espectador à fila (*queueAddCommand*), etc. Os parâmetros do *Config.cfg* e suas funções são apresentados abaixo:

- **queueFile:** Aponta o caminho de onde está o arquivo JSON que contém a fila que será manipulada pelo *chatbot*.
- **botUsername:** Nome do usuário da Twitch que será utilizado como *chatbot* no chat.
- **botOAuth:** *Hash* de autorização gerada pela Twitch para o usuário que será utilizado como *chatbot* poder se conectar ao chat.

[4]https://pt-br.reactjs.org/docs/hooks-intro.html

- **channel:** Nome do canal no qual o *chatbot* irá se conectar para receber os comandos.
- **queueSubscriberPriority:** Número de posições na fila que um inscrito tem de vantagem em cima de um espectador comum. Caso o número seja menor que zero, o inscrito não tem vantagem ao entrar na fila.
- **queueOpen:** Variável que indica se o *chatbot* irá começar com a fila aberta ou fechada ao inicializar.
- **rewardId:** Código único para utilização da opção de gasto de pontos do canal.
- **queueAddCommand:** Configuração do nome do comando para adicionar um usuário à fila.

A Figura 4.4 exemplifica uma configuração do arquivo *Config.cfg*.

```
module.exports = function(){
    this.queueFile = './react/dashboard/public/data.json';

    this.botUsername = 'SFVReplayBot';
    this.botOAuth = 'oauth:hoco8tmt8jgpfhwy4safsdnp123abc';

    this.channel = 'Enkimaru';

    this.queueSubscriberPriority = 2;
    this.queueOpen = true;

    this.rewardId = '';

    this.queueAddCommand = "!adicionar";
};
```

Figura 4.4: Tela de exemplo de configuração do arquivo Config.cfg.

4.3.2 Data.json

É o arquivo que tanto o *chatbot* quanto a *dashboard* utilizam para persistir a condição da fila. Um exemplo de fila armazenada no arquivo *Data.json* é exemplificado pela Figura 4.5. Cada usuário tem os seguintes atributos:

- **id:** Identificador único para cada usuário na fila.
- **username:** Nome do usuário na Twitch.
- **info:** Informação adicional que o *streamer* pode solicitar, como por exemplo, um identificador de replay, código de um estágio ou nome de um personagem dentro do jogo.
- **subscriber:** Atributo que identifica se o espectador é inscrito no canal ou não.

```
[
    {
      "id": "e0576b88-f322-438c-b1dc-57ad7758d683",
      "username": "espectador1",
      "info": "ABCABCABC",
      "subscriber": true
    },
      {
      "id": "e0576388-f322-438c-b1dc-57ad7758d682",
      "username": "espectador2",
      "info": "123123123",
      "subscriber": false
    },
      {
      "id": "e0576288-f322-438c-b1dc-57ad7758d681",
      "username": "espectador3",
      "info": "123456789",
      "subscriber": false
    }
  ]
```


Figura 4.5: Tela de exemplo de uma fila armazenada no arquivo Data.json.

Foram demonstrados neste Capítulo, através de um diagrama de casos de uso, os atores e pacotes que descrevem as funcionalidades propostas do Enkibot. Também foram descritos a arquitetura dos módulos do projeto e os arquivos utilizados para o armazenamento de dados.

5 PRODUTO

Neste Capítulo serão apresentadas as formas de instalação e inicialização do Enkibot, além de exibir os comandos disponíveis para os usuários.

5.1 Instalação

Inicialmente, é necessário realizar o *download* do Enkibot através do link do Github: `https://github.com/Enkimaru/Enkibot`. É possível baixar o repositório do projeto em formato *.zip* ou cloná-lo pelo comando $git\ clone$.

Para a instalação e inicialização do Enkibot é necessário que o *Node.js* e o *React* estejam previamente instalados para que tanto o *chatbot* quanto o *dashboard* possam inicializar. Na pasta do projeto existe um executável chamado *Helper.exe*, utilizado para poder enviar os dados da fila à alguma janela. Caso a escolha seja rodar a versão não-compilada do Enkibot, será necessário a instalação do componente *AutoHotkey*.

5.2 Execução

O primeiro passo é configurar o arquivo *config.cfg* na pasta raiz do projeto com as informações da conta do *streamer* que irá se conectar ao chat da Twitch. Sendo as informações principais: o *nome do canal*, a *conta do próprio chatbot* e o *token* de autorização *OAuth*, disponibilizado pela própria Twitch. É altamente recomendado que a conta usada no Enkibot seja uma conta à parte, ou seja, uma outra conta criada específica para o chat, enquanto a conta do *streamer* fique reservada somente para o seu canal da Twitch.

Para inicializar o Enkibot, é necessário abrir o terminal na pasta raiz e digitar o comando **npm run enkibot**. Com a execução do comando inicializa o *chatbot* na porta 8080 e o *dashboard* na porta 3000.

5.3 Comandos

O *chatbot* pode ser utilizado através de comandos enviados ao chat da Twitch. Todos os comandos começam com um ponto de exclamação (**!**) e só são ativados pelo *chatbot* caso o usuário tenha a permissão necessária. As permissões são divididas em três níveis: **Espectador**, **Inscrito** e **Streamer**.

5.3.1 Espectador

O Espectador é qualquer usuário que esteja assistindo a transmissão do *streamer*. Ele pode usar comandos básicos para participar da fila, dentre eles:

- **!adicionar:** Comando para adicionar o próprio usuário a fila, podendo receber uma mensagem além do comando. Esta mensagem pode conter uma informação extra que o *streamer*, posteriormente, pode usar para identificar algum *replay* ou fase. Caso a configuração *rewardId* esteja configurada, o espectador só poderá adicionar a informação à fila utilizando os pontos do canal e clicando, explicitamente, no botão de resgate de recompensas - no próprio chat da Twitch. Caso o usuário já esteja na fila, ao utilizar o comando, sua posição não é alterada, mas a mensagem (utilizada como parâmetro opcional) é alterada pela última escrita pelo espectador.
- **!fila:** Comando para exibir a fila inteira no chat da Twitch.
- **!sair:** Remove o próprio usuário da fila.
- **!pos:** Exibe no chat da Twitch a posição atual do usuário na fila.

5.3.2 Inscrito

Inscrito é um usuário que apoia o canal com uma assinatura recorrente. Ele utiliza os mesmos comandos de um espectador, mas tem vantagens. Ao utilizar o comando de se adicionar à fila, ele não precisa utilizar pontos do canal e dependendo da configuração de *queueSubscriberPriority*, dispõe de uma prioridade na fila sob os demais espectadores não-inscritos.

5.3.3 Streamer

O *streamer* é o administrador do Enkibot. Além dos comandos básicos, o chatbot disponibiliza os seguintes comandos para o gerenciamento das filas de espectadores:

- **!adicionar user:** Caso o *streamer* utilize o parâmetro *user*, ele pode passar o identificador da conta de outro usuário da Twitch para adicioná-lo à fila.
- **!abrir:** Este comando é utilizado para abrir a fila, permitindo com que novos usuários sejam adicionados à fila.
- **!fechar:** Fecha a fila, não permitindo que outros usuários sejam adicionados ao utilizar o comando **!adicionar**, porém não deleta os usuários já cadastrados na fila.
- **!limpar:** Remove todos os usuários da fila.
- **!carregar:** Carrega uma fila a partir de um arquivo JSON.
- **!prox:** Remove o usuário do topo da fila e o exibe no chat.

Este Capítulo explicitou a instalação e execução do Enkibot e demonstrou, em forma lista, os comandos disponíveis para a utilização do *streamer* e de seus espectadores (inscritos ou não).

6 ENKIBOT

Este Capítulo ilustra um exemplo de utilização do Enkibot, a partir de sua inicialização pelo *streamer* e a utilização de seus comandos, inclusive por outros usuários. O *streamer* utiliza o Enkibot para organizar uma fila de *replays* entre os espectadores.

A inicialização tanto do *chatbot* quanto da *dashboard* - onde será exibida a fila - é feita pelo comando **npm run enkibot**. Este comando deve ser executado pelo terminal dentro da pasta raiz do projeto, conforme ilustrado pela Figura 6.1.

```
pcalado in Enkibot on ?develop [✘!]
❯ npm run enkibot

> enkibot@1.0.0 enkibot
> concurrently "node app.js" "cd react/dashboard && npm start"

[0] Server started on port 8080
[1]
[1] > lamaadmin@0.1.0 start
[1] > react-scripts start
[1]
[0] [23:19] info: Connecting to irc-ws.chat.twitch.tv on port 443..
[0] [23:19] info: Sending authentication to server..
[0] [23:19] info: Connected to server.
[0] [23:19] info: Executing command: JOIN #enkimaru
[0] [23:19] info: Joined #enkimaru
```

Figura 6.1: Tela do terminal exibindo o comando de inicialização do Enkibot.

O *streamer* com o nome de usuário 'Enkimaru' inicia uma *stream* do jogo *Street Fighter V* com o intuito de realizar análises de *replays* das partidas dos seus espectadores. Para poder analisar um replay, o *streamer* necessita que o espectador disponibilize um código de 9 caracteres hexadecimais, o identificador do *replay*.

O usuário ao acessar a página do seu canal pode observar ao lado esquerdo sua transmissão ao vivo e ao lado direto encontra-se o chat, onde serão digitados os comandos ao Enkibot, conforme a Figura 6.2 ilustra.

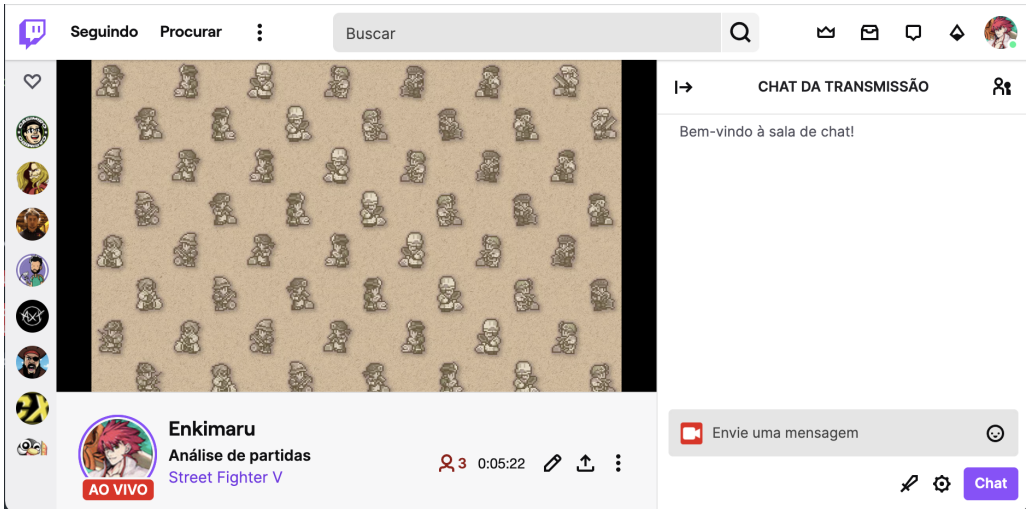

Figura 6.2: Tela da página da Twitch pela visão do streamer.

Primeiramente, o *streamer* digita o comando de exibição da fila **!fila** no chat da Twitch, tendo como retorno do Enkibot uma fila de replays vazia, como mostra a Figura 6.3. Além disso, a Figura 6.4 exibe uma *dashboard* vazia.

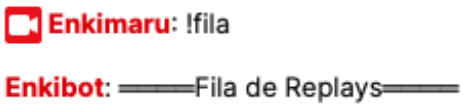

Figura 6.3: Chat da Twitch com o comando de exibição de fila com uma fila vazia.

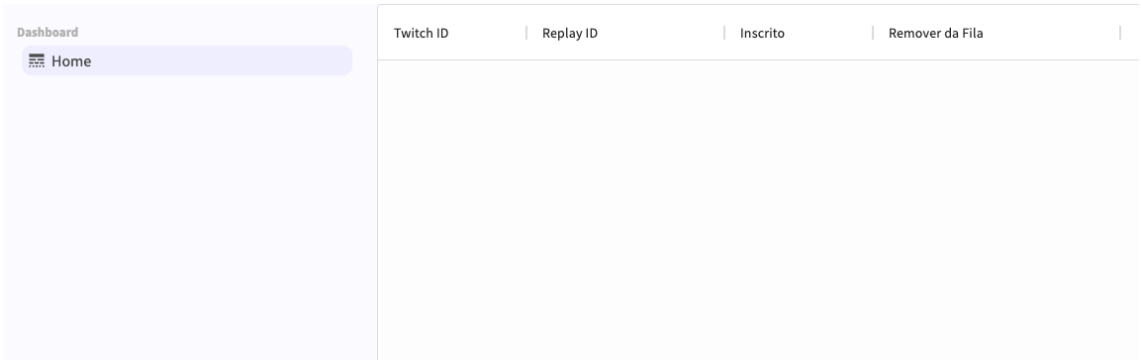

Figura 6.4: Dashboard exibindo uma fila vazia.

Em seguida, a Figura 6.5 exibe o usuário 'Espectador123' que digita no chat o comando **!adicionar** com o parâmetro *123123123*, para ser adicionado à fila.

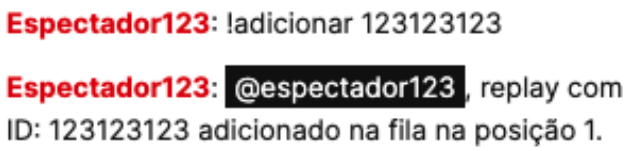

Figura 6.5: Chat da Twitch com o comando para adicionar o usuário 'Espectador123' à fila.

As Figuras 6.6 e 6.7 exibem, respectivamente, a tela do chat da Twitch mostrando o comando executado pelo *streamer* que exibe a fila com apenas um usuário e a *dashboard* com a mesma fila.

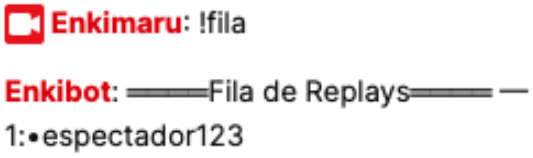

Figura 6.6: Streamer utilizando o comando de exibição de fila no chat da Twitch, exibindo o usuário que foi adicionado à fila.

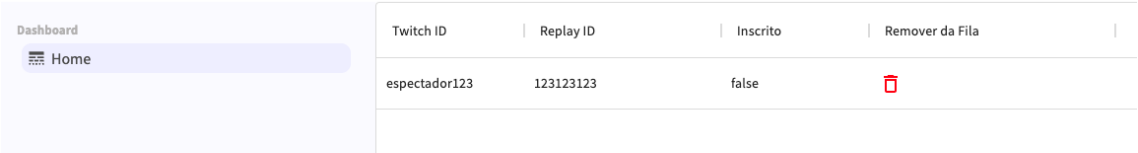

Figura 6.7: Dashboard exibindo o usuário que foi adicionado à fila.

Agora um novo usuário chamado 'asfaltita' digita o comando para adicionar um *replay* à fila com um parâmetro diferente: *abcabcabc*, e com isso é adicionado ao final da fila na posição número 2, conforme Figura 6.8.

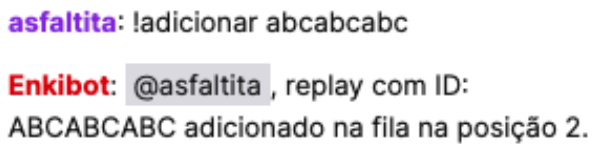

Figura 6.8: Outro usuário utilizando o comando para adicionar um replay na fila.

A Figura 6.9 mostra novamente o comando **!fila** no chat e a Figura 6.10 mostra o estado da *dashboard*, ambos exibindo o espectador novo na fila, por ordem de chegada.

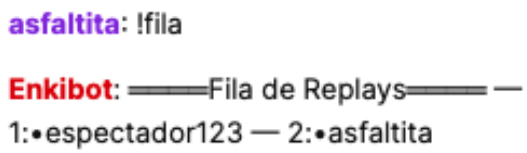

Figura 6.9: Chat da Twitch exibindo 2 usuários na fila de espera.

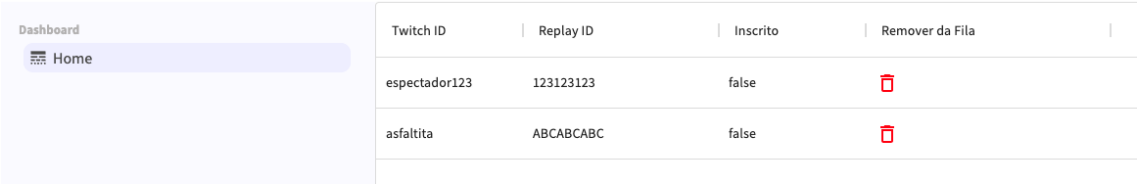

Figura 6.10: Dashboard exibindo os 2 usuários na fila de espera por ordem de chegada.

Um terceiro espectador chamado 'inscrito456' que, diferente dos outros dois usuários, é inscrito do canal, também utiliza o comando de adição de *replays* à fila. Porém, conforme mostra a Figura 6.11, ele não é adicionado ao final da fila, mas sim no inicio, devido à prioridade existente à inscritos.

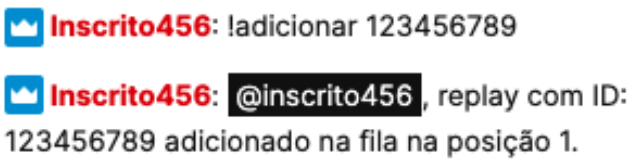

Figura 6.11: Chat da Twitch exibindo o usuário inscrito sendo adicionado ao inicio da fila.

As Figuras 6.12 e 6.13 exibem a fila com essa ordenação prioritária tanto no chat da Twitch quanto na *dashboard*.

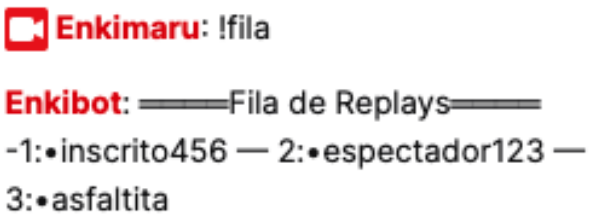

Figura 6.12: Chat da Twitch mostrando a fila com os 3 usuários.

Um dos espectadores utiliza o comando **!pos** para saber sua posição na fila, exemplificado na Figura 6.14.

O *streamer*, então, chama o próximo usuário da fila através do comando **!prox**, fazendo com que o usuário seja chamado pelo Enkibot no chat da Twitch, conforme mostrado na Figura 6.15. Por conta disso, o espectador é removido da fila, como pode ser visto no *dashboard* da Figura 6.16.

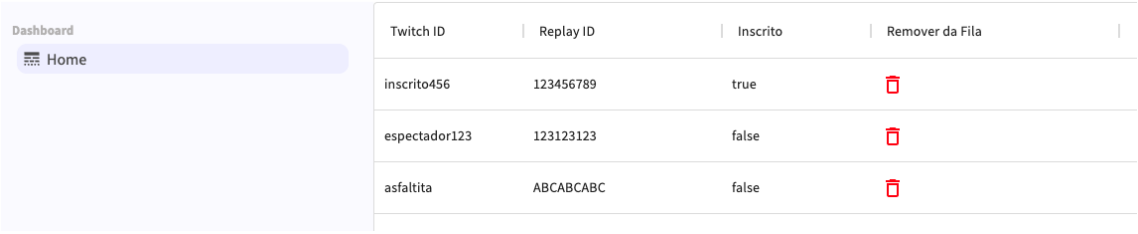

Figura 6.13: *Dashboard* mostrando a fila com os 3 usuários.

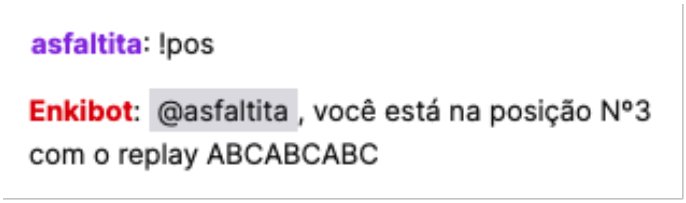

Figura 6.14: Usuário usando comando para descobrir sua posição na fila no chat da Twitch.

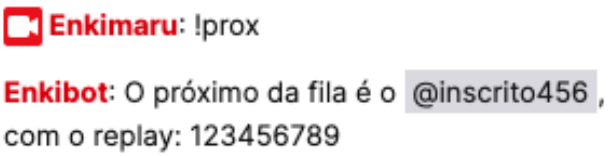

Figura 6.15: Streamer utilizando o comando para chamar o próximo da fila.

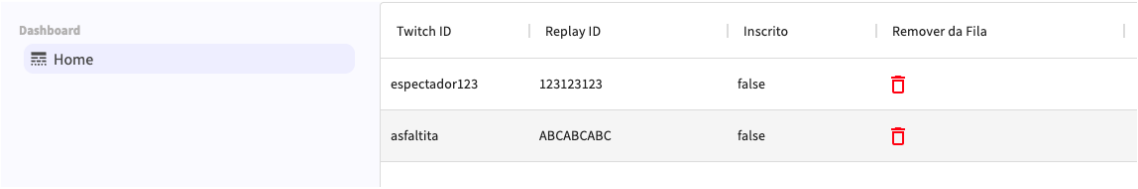

Figura 6.16: Dashboard mostrando os 2 usuários restantes.

O próximo passo do *streamer* é fechar a fila para impedir que novos espectadores se juntem a ela, enviando comando **!fechar** no chat da Twitch, como mostra a Figura 6.17. Com isso, mesmo que outro usuário tente se juntar à fila, o Enkibot não permite a adição de novo usuários à fila, ilustrado pela Figura 6.18.

Figura 6.17: Streamer utilizando o comando de fechar a fila.

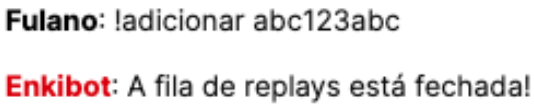

Figura 6.18: Outro usuário utilizando o comando de adicionar replay, mas sendo barrado pois a fila está fechada.

Por fim, o *streamer* utiliza o comando **!limpar** para remover os usuários restantes da fila, fazendo com que tanto o Enkibot quanto a *dashboard* exibam uma fila vazia novamente, como mostrado pelas Figuras 6.19 e 6.20.

Figura 6.19: Streamer utilizando o comando de limpar a fila e em seguida o de mostrar a fila, com ela estando vazia.

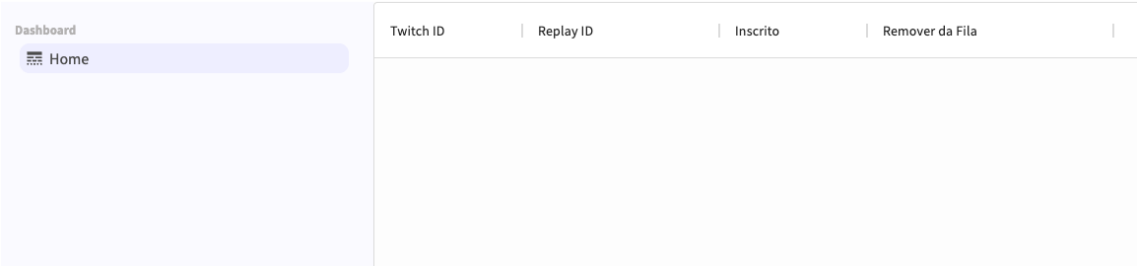

Figura 6.20: *Dashboard* com a fila vazia novamente.

Este Capítulo teve a intenção de demonstrar um exemplo de utilização do Enkibot, desde sua inicialização até o fim de seu uso. Demonstrou também, a utilização de comandos para adicionar e remover usuários à fila e como o *dashboard* se comportava em cada momento que havia uma alteração na fila.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho teve a motivação de atacar uma necessidade encontrada na plataforma de *streaming* da Twitch, que ao longo dos anos vem adicionando novas ferramentas de apoio aos *streamers*, porém ainda negligencia o gerenciamento de filas no que tange a ordem das solicitações dos espectadores. Assim, este trabalho de conclusão de curso propôs um chatbot para a Twitch, denominado Enkibot, desenvolvido em Node.js que se conecta ao chat do canal de um *streamer* e que, por meio de comandos disparados no próprio chat, é capaz de gerenciar as filas de prioridade. Além disso, para uma melhor visualização das filas, o Enkibot disponibiliza um *dashboard* construído em React e, caso necessário, oferece ainda um *script* em AutoHotkey que permite enviar informações das filas criadas no chatbot a outras aplicações. Somado a isso, o Enkibot é um software *open-source* e colaborativo, possibilitando o desenvolvimento de novos comandos, além de disponibilizar configurações que em outros trabalhos só poderiam ser obtidos mediante planos pagos de assinatura. Por outro lado, o chatbot proposto apresenta algumas restrições, sendo elas: (1) exigir como pré-requisito ao Enkibot a instalação de dependências na máquina do *streamer*, ou seja, o download e instalação dos frameworks Node.js e React e; (2) o *script* desenvolvido em AutoHotkey ser compatível somente em versões do sistema operacional Windows.

Para possíveis trabalhos futuros, pode ser estudado a utilização de um outro framework para encapsular e disponibilizar o *bot* de forma única, como, por exemplo, através do Electron[1] para remover a necessidade de exigir do *streamer* a instalação prévia de dependências para o completo funcionamento do Enkibot. Outro ponto interessante seria adaptar os comandos para uma versão editável, ou seja, criar um arquivo de configuração que permita mudar o nome de cada um, permitindo inclusive a tradução dos comandos do Enkibot para outros idiomas. Em relação ao *dashboard*, seria interessante também adicionar novas funcionalidades como permitir a alteração da ordem na fila ou ser capaz de inserir novos usuários diretamente da página, facilitando o trabalho do administrador/*streamer* do Enkibot.

Outra tecnologia a ser analisada seria a integração com a plataforma Discord[2]. Poderia ser aproveitado o chat da plataforma de forma semelhante ao da Twitch, permitindo que os usuários enviem comandos diretamente pelo chat do Discord. O *streamer* poderia, por exemplo, organizar uma fila antes que sua própria transmissão começasse ou limitar comandos por grupos no próprio canal do Discord.

[1] Definição em: https://www.electronjs.org/pt/docs/latest

[2] Plataforma de chat por voz e mensagens instantâneas. Disponível em: https://discord.com/

REFERÊNCIAS

ALVES, R. Disponível em: https://www.movavi.com/pt/learning-portal/plataformas-de-streaming-de-jogos.html. Acesso em: 28 novembro 2021.

AUTOHOTKEY. **Documentação AHK**. Disponível em: https://www.autohotkey.com/#keyfeatures. Acesso em: 16 novembro 2021.

BOLLATI, E. Disponível em: https://estnn.com/pt/now-live-streaming-full-time-on-twitch/. Acesso em: 5 dezembro 2021.

CHEN, S.; CHEN, Z.; ALLAIRE, N.; BEALL, M.; BELMAS, G.; CHRISTIANS, C.; CORR, M.; DAGGETT, C.; LARSON, B.; LEGEWIE, N. et al. **Legal and Ethical Issues of Live Streaming**. [S.l.]: Lexington Books, 2020.

CHURCHILL, B. Disponível em: https://github.com/SirSkaro/Twitch-Queue-Bot/. Acesso em: 10 setembro 2021.

CHURCHILL, B. Disponível em: https://github.com/SirSkaro/Twitch-Queue-Bot/issues/3. Acesso em: 10 setembro 2021.

DARE DROP. Disponível em: https://daredrop.com/blog/the-best-streamer-tools-extensions-plugins/. Acesso em: 5 dezembro 2021.

DUNHAM KEN; MELNICK, J. **Malicious bots** : an inside look into the cyber-criminal underground of the internet. [S.l.]: Boca Raton : CRC Press, 2009. v.1.

FINGAS, J. **Twitch viewership more than doubled over the last year**. Disponível em: https://www.engadget.com/streamlabs-twitch-viewership-doubled-in-one-year-150003530.html. Acesso em: 30 novembro 2021.

FOSTER, J. **GitHub tmi.js**. Disponível em: https://github.com/tmijs/tmi.js. Acesso em: 16 novembro 2021.

GOUTHERAUD, P. Disponível em: https://dashboard.twitch.tv/extensions/hvvijr100w490kxs6y5tj7twac5hh0. Acesso em: 10 setembro 2021.

HEMEL, Z. **Facebook's React JavaScript User Interfaces Library Receives Mixed Reviews**. Disponível em: https://www.infoq.com/news/2013/06/facebook-react/. Acesso em: 16 novembro 2021.

JAKU. Disponível em: https://warp.world/v1/multi-queue. Acesso em: 10 setembro 2021.

JOHNSON, M.; WOODCOCK, J. The impacts of live streaming and Twitch.tv on the video game industry. **Media Culture & Society**, [S.l.], 12 2018.

KHAN, R.; DAS, A. **Build Better Chatbots**: a complete guide to getting started with chatbots. [S.l.]: Apress, 2017.

KINAST, P. Disponível em: https://www.oficinadanet.com.br/youtube/36276-twitch-vs-youtube. Acesso em: 28 novembro 2021.

LI, Y.; WANG, C.; LIU, J. A Systematic Review of Literature on User Behavior in Video Game Live Streaming. **International Journal of Environmental Research and Public Health**, [S.l.], v.17, n.9, 2020.

MORRIS, T. **Twitch For Dummies**. [S.l.]: Wiley, 2019.

NODEJS. **Documentação Node.js**. Disponível em: https://github.com/nodejs/node/blob/master/README.md. Acesso em: 10 novembro 2021.

OLSTON, C.; NAJORK, M. **Web Crawling**. [S.l.]: Now Publishers, 2010. (Foundations and trends in information retrieval).

PANCHADAR, A. Disponível em: https://reut.rs/3xTsely. Acesso em: 4 dezembro 2021.

PATEL, N. **Definição Bots**. Disponível em: https://neilpatel.com/br/blog/bots-estrategia-de-marketing-digital/. Acesso em: 27 novembro 2021.

PAZOS, A. Disponível em: https://medium.datadriveninvestor.com/chatbots-how-they-save-a-business-time-and-money-bb4427697efb. Acesso em: 16 dezembro 2021.

PEREIRA, C. **Aplicaçoes Web Real-time Com Nodejs**. [S.l.]: CASA DO CÓDIGO, 2014.

POTENZA, A. Disponível em: https://netshow.me/blog/live-streaming-tudo-o-que-voce-precisa-saber#O_que_e_live_streaming. Acesso em: 27 novembro 2021.

RAJ, S. **Construindo Chatbots com Python**: usando natural language processing e machine learning. [S.l.]: Novatec Editora, 2019.

REACT. **Documentação React**. Disponível em: https://pt-br.reactjs.org/. Acesso em: 16 novembro 2021.

SANTANA, L. Disponível em: https://www.conteudoinboundmarketing.com.br/live-streaming/. Acesso em: 27 novembro 2021.

SJöBLOM, M.; HAMARI, J. Why do people watch others play video games? An empirical study on the motivations of Twitch users. **Computers in Human Behavior**, [S.l.], v.75, p.985–996, 10 2017.

SUCHACKA, G.; IWAńSKI, J. Identifying legitimate Web users and bots with different traffic profiles–an Information Bottleneck approach. **Knowledge-Based Systems**, [S.l.], v.197, p.105875, 04 2020.

SUGGITT, C. **TheGrefg smashes most Twitch stream viewers record**. Disponível em: https://www.guinnessworldrecords.com/news/2021/3/thegrefg-smashes-most-twitch-stream-viewers-record-653818. Acesso em: 30 novembro 2021.

TWITCH. Disponível em: https://www.twitch.tv/bits. Acesso em: 28 novembro 2021.

TWITCH. Disponível em: https://help.twitch.tv/s/article/channel-points-guide?language=pt_BR. Acesso em: 28 novembro 2021.

TWITCH. Disponível em: https://www.twitch.tv/p/pt-br/legal/bits-acceptable-use/. Acesso em: 28 novembro 2021.

YEOW, C. **Firefox Secrets**. [S.l.]: SitePoint, 2005. (Need-to-know guide).

YOUTUBE. Disponível em: https://support.google.com/youtube/answer/7288782. Acesso em: 28 novembro 2021.